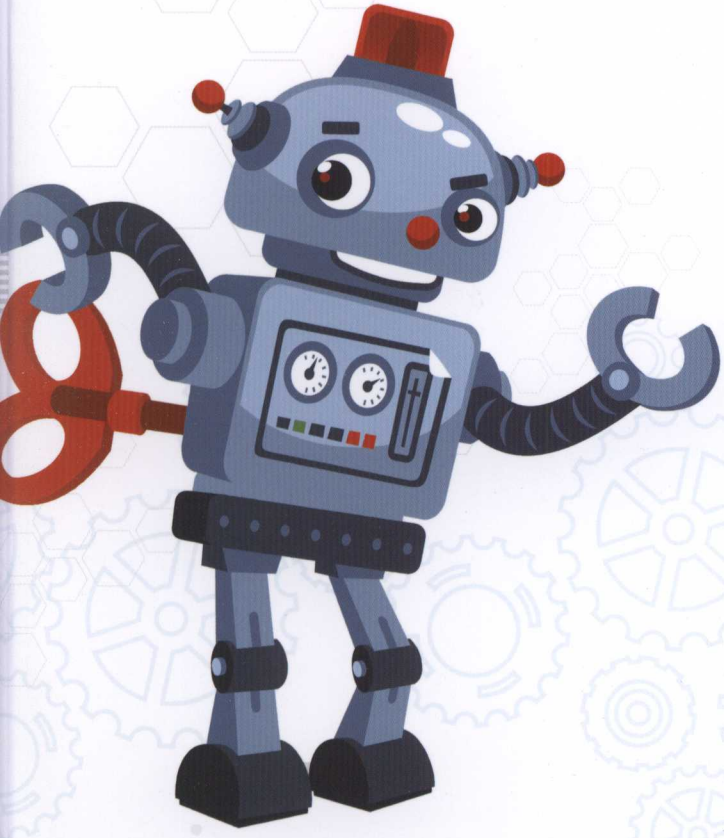
Top Quant
CHRD前海智库
CHIAN HIGH RESEARCH AND DEVELOPMENT THINK TANK
公共立场 · 民间智慧

· 青少年学编程系列丛书 ·

机器人 Python

极客编程入门与实战

Python极客团队 著



Python是一门学习简单、功能强大、
可以终身受益的编程语言，
也是成功迈向智能机器人的基石。

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Python极客团队（Python Geek Team，简称PGT），是专注于中国Python极客领域、Python智能硬件领域的开放性技术合作团队。

PGT团队工作内容包括：

- Python极客软件、硬件开发，如开发套件、无人机、机器人、人工智能、机器学习、物联网等。
- 跟踪、收集、统计Python硬件领域的行业信息和最新科技动态。
- 促进联盟成员在技术、市场、知识产权等领域的交流合作与自律，协同推进国内Python极客领域和相关产业链的有序发展。
- 大力推动Python智能硬件领域与用户行业之间的深入合作，加速相关技术与产品在各行业中的普及应用。

Python极客团队网址：www.zroboto.com
www.ziwang.com

QQ群：419523710（python极客）

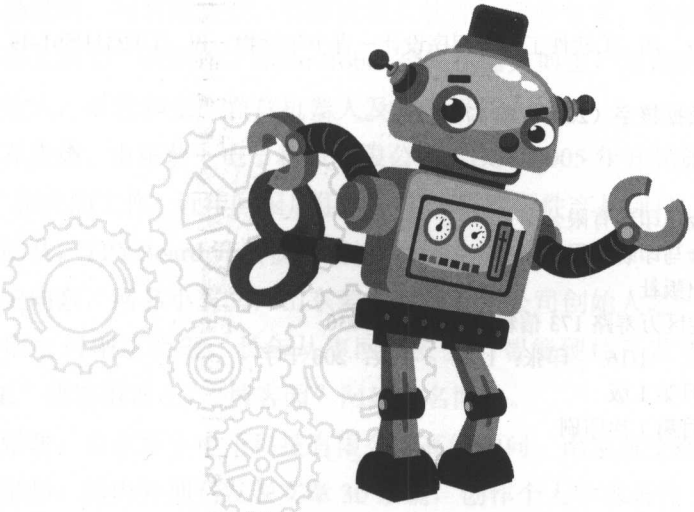
出版编辑联络：黄爱萍
微信／QQ：69476637
邮箱：huangaip@phei.com.cn

·青少年学编程系列丛书·

机器人 Python

极客编程入门与实战

Python极客团队 著



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

Python 是一种面向对象、解释型的计算机程序设计语言，其简洁实用、高效，拥有众多模块库，可移植，跨平台，简单易学，易于上手。随着计算机深入发展，Python 逐渐成为最适合青少年做创意设计的语言，也同样适合 Geek（极客）、创客等针对智能硬件的设计与编程。极客中有一句名言：需要的就是创新和新奇，盲目地跟从和愚昧是不可原谅的。同样，青少年做创意设计需要灵感与创新，从简单的方案入手，更能发挥青少年的创新意识。

本书首先通过介绍 Python 的简单入门案例，让读者了解和熟悉 Python 的基础语法结构，以及 Pandas 等绘图风格。通过对 MicroPython 和 PyBox、PyMini 等软件、硬件平台的介绍，以及大量精心挑选的简单有趣、实用性强的实际案例，如 GPIO 控制、LED 灯管、机器人舵机、智能小车等，增加青少年的动手能力，让广大青少年、初学者，通过简单学习快速掌握 Python 基础编程，为进一步学习机器人编程奠定扎实的基础。相信读者在本书中能体会到 Python 语言的简洁、智能硬件设备编程的趣味，以及亲手设计作品的灵感。

图书在版编目（CIP）数据

机器人 Python 极客编程入门与实战 / Python 极客团队著. —北京：电子工业出版社，2017.9
（青少年学编程系列丛书）
ISBN 978-7-121-32292-1

Ⅰ. ①机… Ⅱ. ①P… Ⅲ. ①软件工具－程序设计－青少年读物 Ⅳ. ①TP311.561-49

中国版本图书馆 CIP 数据核字（2017）第 176718 号

责任编辑：黄爱萍
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：720×1000　1/16　印张：12.5　字数：200 千字
版　　次：2017 年 9 月第 1 版
印　　次：2017 年 9 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

丛书编委会

杜军，Lapsule Inc.（北京锐和信科技有限公司）联合创始人、CTO，Pythoner、前 iOS 资深技术；MicroPython 社区贡献者，MicroPython.cn 社区维护者，lapos（基于音乐和语音的 IoT OS）主要维护者，专注于 MicroPython 在音乐、IoT 领域的探索。

邹研明，网名 amida，壹本堂创始人，主要从事 0～21 岁全人格内驱成长教育课题研究。壹本堂是国内首批少年创客项目发起机构，开发有机器人控制器及全系列产品、完整课程体系、赛事组织体系和师资培训体系，先后承办 2016 届、2017 届 LEGO（乐高）机器人全国巡回赛事。

余勤，Gene.Yu，上海大学微电子中心硕士，AMD 验证工程师，参与多个 GPU 项目，精通硬件视频编解码、芯片验证，有留片经验，熟悉整个数字芯片设计流程，擅长数据分析，热爱 Python 量化分析，极宽 Top 开源团队发起成员。

蔡磊，武汉大学信息专业学士，原高通公司工程师，项目经理。精通 Python 数据挖掘、数据库技术、机器学习和量化交易理论。熟悉 4G 无线技术、手机芯片架构以及 VOIP 技术，极宽 Top 开源团队核心成员。

王硕，网名信平，北京科技大学计算机系毕业，高级软件工程师，精通 Python 数据分析，擅长 Java、JavaScript、HTML 5 和数据库技术，熟悉量化交易理论、互联网、移动应用，著有《PyQT5 快速开发与实战》，极宽 Top 开源团队核心成员。

前　言

Python 是一门学习简单、功能强大并可以终身受益的编程语言。

阿尔法狗、围棋大师、机器学习、人脸识别、金融量化、数据分析……都可以运用 Python 实现，Python 似乎无所不能。

2016 年 11 月，全球领先的 FPGA 开发商德致伦（Digilent）公司，率先发布了 PYNQ 开发板，可直接使用 Python 语言进行 FPGA 准芯片级硬件编程，学术价格仅 65 美元。

也许 FPGA 和芯片设计对于广大公众来说有些遥远，但这几年风靡全球的“创客”运动，其鼻祖就源自小小的“树莓派”（Raspberry Pi）。

尽管树莓派和创客运动已经非常成功，但其解决的还只是表层的硬件问题，更关键的软件层面、程序开发、软硬一体化、智能控制等方面始终未能突破。其背后原因在于树莓派及其凭借的传统汇编、C 语言的开发平台，缺乏如今互联网时代、大数据时代的大数据、人工智能模块库，所以很多工作都要从零开始。

Python 已经成为人工智能、数据分析等领域事实上的工业标准编程语言，Python 的硬件衍生版本 MicroPython 已经从概念上成为智能开发、物联网应用的工业级编程语言。

幸运的是，国内有关企业、技术团队，在最新一轮的智能化硬件平台军备大赛中没有被淘汰，而是紧跟国际技术前沿，在国内迅速完成了多个不同版本的硬件开发平台设计，组建了多个相关的技术社区。

- PyBox 开发套件：http://www.zroboto.com。
- PyMini 开发套件：http://www.zroboto.com。
- TPYBoard 开发板：http://www.tpyboard.com。
- PYB-Nano 迷你开发板，MicroPython 中文社区：http://www.mimcropython.org.cn。
- Newbit 开发板，MicroPython 中文社区：http://www.mimcropython.org.cn。

很多原本烦琐的智能化设计，例如人脸识别、车牌识别等，在使用全新的 Python 开发板和各种人工智能模块库时，仅需数十行代码就可以实现。

更加令人期待的是，被誉为“黑科技”的谷歌 TensorFlow 神经网络平台，首选的开发语言也是 Python。

有了先进的软件、硬件开发平台，剩下的只是创意。

中国人，特别是中国的年轻人，是全球最富有创业、创新精神的一群人，这样的一个群体难道还会缺乏创意吗？

《机器人 Python 极客编程入门与实战》只是“青少年学编程系列丛书”的第一本，本系列包括以下作品。

- 《机器人 Python 极客编程入门与实战》：Python 开发板套件的使用与学习，包括数十个简单入门案例，如 LED 控制、Wi-Fi 控制、机器小车等。
- 《机器人 Python 智能开发与实战》：基于 Python 的智能化机器人开发设计，比如语音识别、电脑绘画等。
- 《机器人 Python 案例汇编》：汇集 Python 极客团队和国内众多一线高手设计的各种实用、经典智能案例。

“青少年学编程系列丛书”只是“Python 极客项目”的起点，也是新一代智能化硬件的起点，我们期待更多的同行、更多的年轻人加入这个领域。

本书所有案例程序可用于 zwPython 平台，以及各种支持 Python 3 的设备平台，包括 Linux 操作系统、iOS 系统，以及安卓系统等。

其他非 zwPython 用户运行本书程序时，如果出现问题，通常是缺少有关的

Python 模块库，可以根据调试信息安装相关的 Python 模块库，再运行相关程序。

zwPython 及本书配套资料下载地址，请参见 Top 极宽量化社区“下载中心”：http://topquant.vip 或 http://ziwang.com。

何海群

北京极宽科技 • www.TopQuant.vip

2017 年 7 月 21 日于中关村创业大街

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：*http://www.broadview.com.cn/32292*

目　录

第一部分　Python 基础

第二部分　PyBox 实战案例

第三部分　智能小车

第四部分　Python-mini 编程案例

第一部分　Python 基础

第 1 章　Python 简介

1.1　入门简单、功能强大

有学者认为：“Python 入门简单、功能强大，从 8 岁到 80 岁都可以学习；小学生、博士生一样可以使用，是真正的终身编程语言。”

Python 是最适合编程初学者的语言，是目前 IT 行业唯一的入门简单、功能强大的工业级开发平台，几乎成为 IT 行业的万能开发平台。

1．入门简单

任何熟悉 JavaScript 脚本、Visual Basic、C 语言、Delphi 的用户，通常一天即可学会 Python。

即使是不会编程的美工设计师、打字员，一周内也能熟练掌握 Python，学习难度绝对不会高于 Photoshop、五笔，至少笔者现在还不会使用五笔字型。

2．功能强大

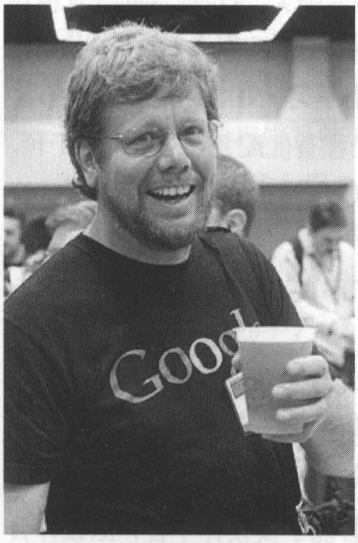
海量级的 Python 模块库，提供了 IT 行业最前沿的开发功能。

- 大数据：Pandas 已经逐步碾压 R 语言。
- 高性能计算 CUDA：Python、与 C（C++）、Fortran 是 NVIDIA 公司官方认可的 3 种编程语言，Python 也是目前唯一适合大众用户的 CUDA 编程工具。
- 机器学习：TensorFlow、PyTorch、Scikit-learn、Theano 都是国际上热门的机器学习平台。
- 自然语言：NLTK 是全球首选的自然语言处理平台；spaCy 是工业级 NLP

平台。

- 人脸识别：OpenCV 有光流算法、图像匹配和人脸算法。
- 游戏开发：Pygame 提供图像、音频、视频、手柄、AI 等全套游戏开发模块库。
- 字体设计：FontForge 是唯一商业级的字体设计开源软件，内置的脚本语言和底层核心的 FontTools，都是基于 Python。
- 电脑设计：Blend、GIMP、Inkscape、MaYa、3ds Max 都内置或扩展了 Python 语言支持。

上述 Pandas、CUDA、TensorFlow、PyTorch、Scikit-learn、Theano 为 Python 模块库或 IT 行业术语。



吉多·范罗苏姆（Guido van Rossum）是一名荷兰计算机程序员，他作为 Python 程序设计语言的作者而被人们熟知。

他对 Python 设计的目标是：

- 一门简单、直观的语言并与主要竞争者一样强大。
- 开源，以便任何人都可以为它做贡献。
- 代码像纯英语那样容易理解。
- 适用于短期开发的日常任务。

既然 Python 如此美好，而且是 100%免费的开源软件，学习 Python 的人也越来越多，为什么 Python 相对于 C#、JavaScript、Visual Basic、C 语言等，始终还只是一种小众语言呢？

笔者认为，Python 的“大众化”之路，存在以下两个瓶颈。

- 配置：软件行业有句俗话“搞懂了软件配置，就学会了一半”。对于 Python 和 Linux 等许多开源项目而言，80%的问题都出现在配置方面，尤其是模块库的配置。
- OOP（面向对象程序设计）：大部分人都认为 Python 是一种“面向对象”

的编程语言，而 OOP 的编程风格，业界公认比较繁杂。

如果能够解决好以上两个问题，Python 的学习难度可以降低 90%，而在应用领域和开发效能方面，则可以瞬间提升数十倍效能，而且这种提升是零成本的。

3. 难度降低，性能提高

Top 极宽开源团队在 WinPython 软件包的基础上，推出了“zwPython”——集成式 Python 开发平台：

- 提出“零配置、零对象”的研发理念，绿色软件封装模式，类似 Mac 开箱即用风格，无须安装，解压即可直接使用，还可以放入 U 盘，支持 Mob-APP 移动式开发编程。
- “外挂”式“核弹”级开发功能，内置很多功能强大、IT 前沿的开发模块库，例如 OpenCV 视觉、人脸识别、CUDA 高性能 GPU 并行计算（OpenCL）、Pandas 大数据分析、TensorFlow、PyTorch 机器学习、NLTK 自然语言处理。
- 便于扩展，用户可以轻松增删相关模块库，全程智能配置，无须用户干预，就像拷贝文件一样简单，而且支持 U 盘移动便携模式，真正实现了“一次安装，随处可用”。
- 针对中文开发文档缺乏、零散的问题，内置多部中文版 OpenCV、FontForge 和 Python 入门教材。
- 大量示例脚本源码，涵盖 OpenCV、CUDA、OpenCL、Pygame 等。

如此种种只是为了便于 IT 行业外的用户能够零起步、快速入门，并且短时间内应用到生产环节中去。

- zwPython 前身是 zw2015sdk：即字王（zw）智能字模设计平台，原设计目标是为广大美工设计师提供一款统一的、可编程的字体设计平台，以便于大家交流。美工设计师、美工都是文艺青年、IT 小白，所以，简单是必需的，开箱即用也必须是标配。
- 做设计，图像处理 PIL、Matplotlib 模块是必需的。
- 集成了 OpenCV 作为图像处理、匹配模块，自然也提供了机器学习功能。
- 字模处理数据量很大，属于大数据范畴，必须集成 SciPy、NumPy 和 Pandas 数据分析模块。

- 由于原生 Python 速度慢，所以增加了 PyCUDA、OpenCL 高性能 GPU 计算模块。

如此一而再、再而三地扩充，发现 zwPython 已经基本覆盖了目前 Python 和 IT 编程 90%的应用领域，因此又增加了部分模块，将 zwPython 扩展成为一个通用的、集成式 Python 开发平台。

4.“零对象”编程模式

虽然很多人认为 Python 是一种“面向对象”的编程语言。但对于初学者而言，把 Python 视为一种 Basic 风格的、过程式入门语言，学习难度可以降低 90%，基本上学习一小时，即可动手编写学习代码。

有人说，“面向对象”最大的好处是方便把人脑子搅乱。

Windows、Linux、UNIX、Mac OS X 内核都是使用 C 语言、汇编写的。有一种系统是 C++写的内核，就是诺基亚的塞班系统，据说代码量比 Windows XP 还大，连他们自己的程序员都无法维护。

“零对象编程模式，用 Basic 的方式学习 Python”，是笔者向 Python 等编程语言的入门用户提出的一种全新的学习理论，一家之言，仅供参考。

“Talk is cheap，Show me the code！”

大家还是多多动手。

“零配置”大家很容易理解，关于“零对象”下面再补充几点。

- 不写“面向对象”风格的代码不等于不能使用，对于各种采用“对象”模式开发的模块库，我们仍然可以直接调用。
- 将 Python 视为非“面向对象”语言并非“大逆不道”，事实上，许多人认为，Python 也是一种类似 LISP 的“函数”编程语言。
- 笔者从事编程十多年，从未用过“面向对象”模式编写过一行“class”（类对象）代码，依然可以应对各种编程工作。
- 目前“面向对象”编程理论，在业界仍然争论不休，入门者功力不够，最好避开强者之间的火力杀伤。
- “面向对象”的鼻祖 C++ 11 标准，直到 2015 年依然处于推广阶段，而且争议纷纷。

- “面向对象”过于复杂，与“人生苦短，我用 Python”的优雅风格天生不合。

1.2 用户运行平台

本节主要讲解 Python 开发环境和数据包的配置、应用流程方面的知识。

本书所有案例程序均采用纯 Python 语言开发，除特别指明外，均默认使用 Python 3 语法，且经过 zwPython 平台测试。

zwPython 是 Top 极宽开源组推出的一个 Python 集成版本，属于免费开源软件。系统内置了数百种专业的 Python 模块库，无须安装，解压即用。有关 zwPython 的使用，可参考软件自带的《zwPython 用户手册》。

本书所有案例程序可用于 zwPython 平台，以及各种支持 Python 3 的设备平台，包括 Linux 操作系统、Mac 苹果电脑，以及安卓系统、树莓派。

其他非 zwPython 用户运行本书程序时，如果出现问题，通常是缺少有关的 Python 模块库，可以调试信息安装相关的 Python 模块库，再运行相关程序。

zwPython 及本书配套下载地址，请参见 Top 极宽量化社区“下载中心”：http://topquant.vip 或 http://ziwang.com。

1.3 程序目录结构

本书配套程序的工作目录是 zwPython\py_demo，这也是本书默认的工作目录，凡是没有特别标注目录的脚本文件，一般都位于该目录。有关的程序会定时在读者群发布更新，请读者及时下载。

相比普通的 Python 版本，本书配套的教学版的 zwPython 目录中多了一个 py_demo 目录。

py_demo 目录收录了相关培训课程的配套代码和所需数据，py_demo 目录也可以复制到其他目录，建议放到 zwPython 根目录下。

zwPython 目录结构中的其他子目录如下。

- \zwPython\doc\：用户文档中心，包括用户手册和部分中文版的模块库

资料。

- \zwPython\py35\：Python 3.5 版本系统目录，除增加、删除模块库外，一般不需要改动本目录下的文件，以免出错。另外，如果日后 Python 版本升级，这个目录也会变化，如 Python 3.6，会采用 py36 作为目录。
- \zwPython\demo\：示例脚本源码。
- \zwPython\zwrk\：zw 工作目录，用户编写的脚本代码文件建议放在本目录下。

1.4　Spyder 编辑器界面设置

1. 开发环境界面设置

在设置界面之前，可随意把一个 Python 源码文件，用鼠标拖到 Python 编程语言编辑器 Spyder 的编辑框中，如图 1-1 所示。

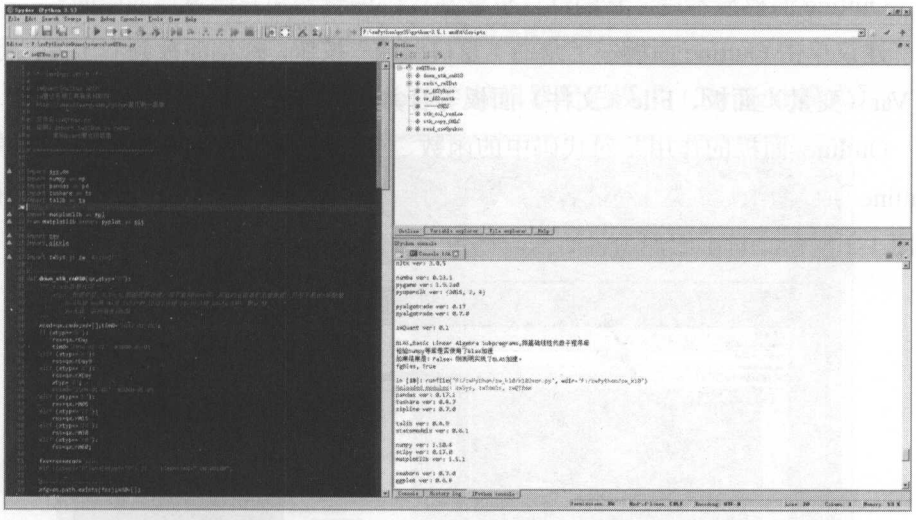


图 1-1　Python 编程语言编辑器 Spyder 编辑框界面

Spyder 编辑器的界面设计非常合理，参考了 MATLAB，特别适合数据分析，很多国际大企业都选择用这种布局作为标配。

通常需要优化的只有 Outline（导航）面板，又称函数列表面板，类似于 Delphi

语言的 Struct 函数列表面板。

在 Spyder 编辑器默认配置中，Outline 面板是不显示的，单击菜单 View→Panes →Outline，如图 1-2 所示，将显示 Outline 面板。

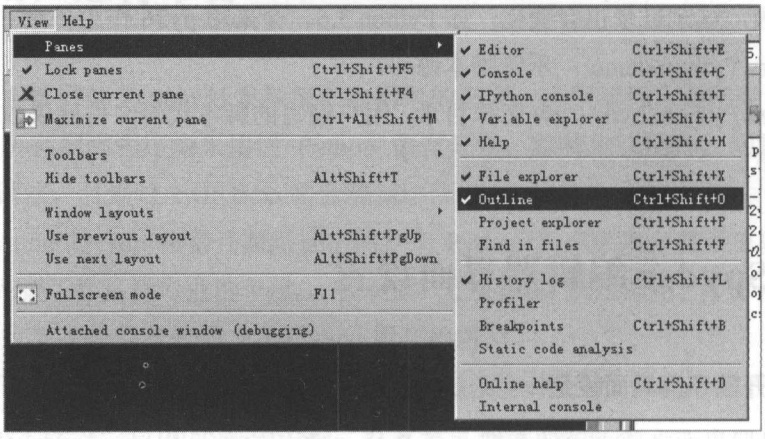


图 1-2　显示 Outline 面板

Outline 面板显示后，它的默认位置在代码编辑器和右侧窗口的中间。

建议单击 Outline 面板左上角的“窗口缩放”按钮，拖动面板到右上方，将其与 Var（变量）面板、File（文件）面板等合并。

Outline 面板的作用是对代码中的函数、类、变量进行快速导航定位。单击 Outline 面板的函数、类、变量名称后，左侧代码编辑器就会自动移动到相关代码，如图 1-3 所示。对于大型项目而言，使用 Outline 面板可以提高效率。

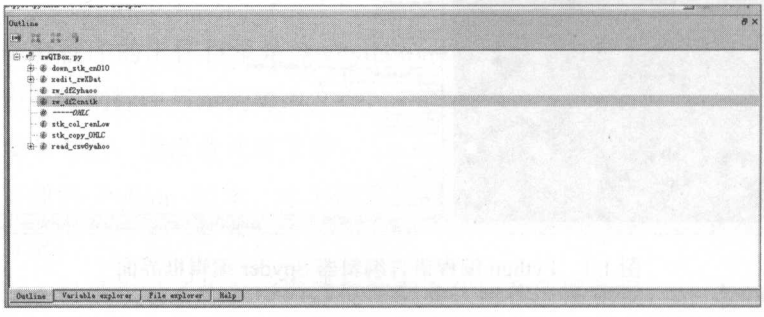


图 1-3　Spyder 编辑器 Outline 面板

需要注意的是，由于 Spyder 软件未来版本将升级，具体操作界面和细节可能

会有所不同，本书其他软件和模块也是如此，这属于正常情况，大家无须担心。

2. 代码配色技巧

zwPython 的 IDE 代码编辑器是 Spyder，默认配色是 Spyder 模式，采用白底黑字，与传统的 IDE 环境差别很大，如图 1-4 所示。

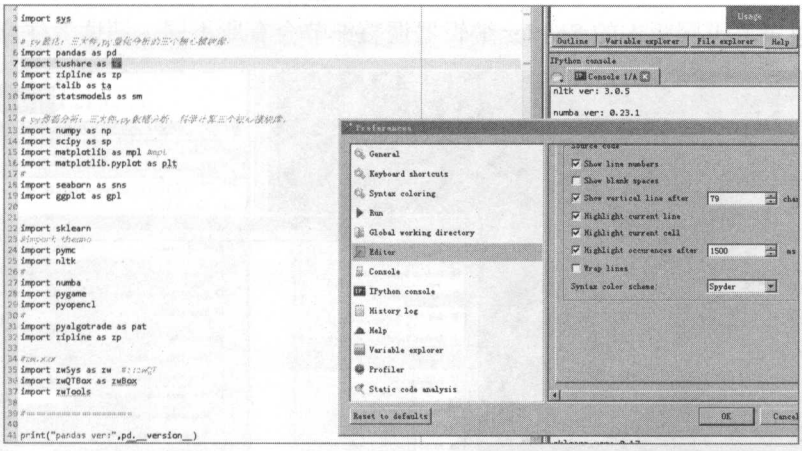


图 1-4　Spyder 编辑器配色模式

如图 1-5 所示是最新的 delphi-xe10 的编辑器配色模式（Twilight 模式）。

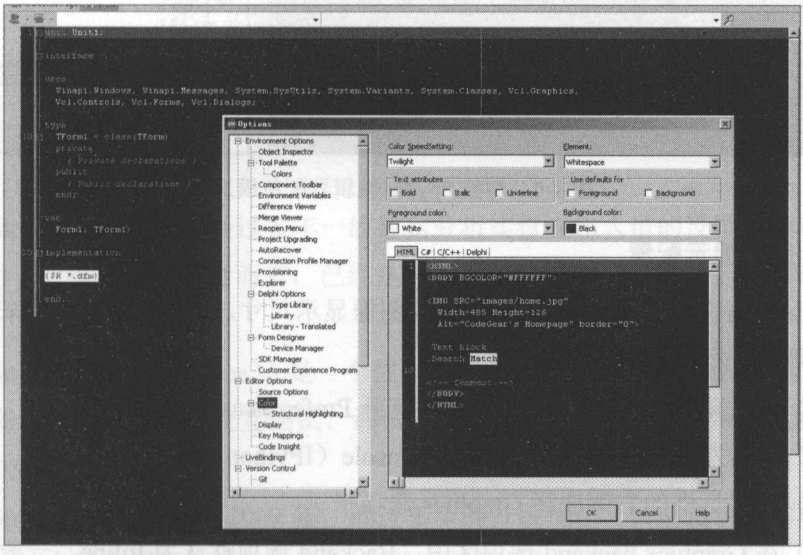


图 1-5　delphi-xe10 编辑器配色模式

这种黑底模式也是微软 VS 等开发平台标准的代码编辑器配色模式。幸运的是，Spyder 编辑器内置的配色模式里也有类似的模式。

运行 Spyder 编辑器，单击菜单 Tools→Preferences，打开 Preferences 对话框。在左侧的列表框中选择 Editor(编辑器)，在右侧 Display(显示)面板的 Syntax color scheme（语法配色方案）下拉列表框中选择 Spyder/Dark（暗调）模式即可，如图 1-6 所示。不同版本的 Spyder 编辑器调整细节会有所不同，请读者注意。

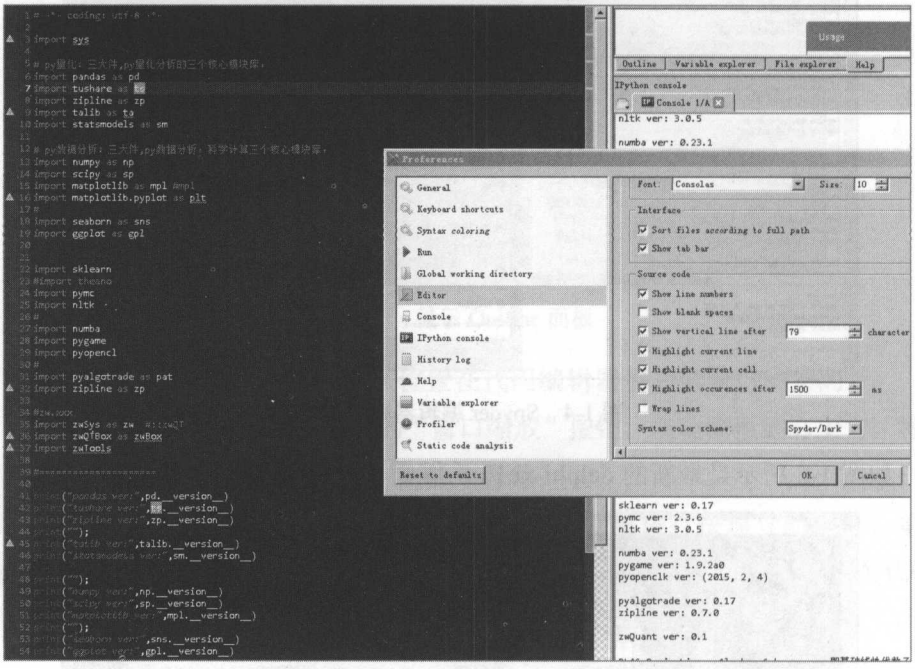


图 1-6　调整 Spyder 编辑器配色模式

3. 图像显示配置

Python 语言的 Spyder 编辑器默认的图像显示尺寸，对于高清显示器来说有些偏小，需要进行调整，具体步骤如下。

（1）单击菜单 Tools→Preferences，打开 Preferences 对话框。

（2）单击左侧列表框中的 IPython console（IPython 控制台）。

（3）在对话框的右侧选择 Graphics 选项卡。

（4）在 Graphics backend 选项区中，Backend 选项默认为 Inline，一般不需要

改，如要进行交互分析，可以设置为 Automatic（自动模式）或者 Qt（Qt 模式）。

（5）在 Inline backend 选项区中可以调整内置图像的大小，默认值 Width 为 8、Height 为 5，建议将 Width 改为 10、Height 改为 6。

此外，建议勾选对话框上部的 Automatically load Pylab and NumPy modules 复选框（会自动加载 Pylab、NumPy 模块），如图 1-7 所示。

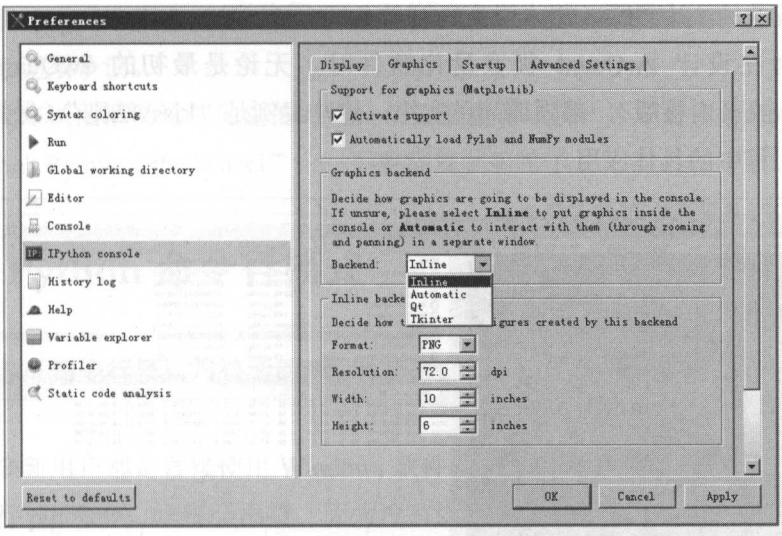


图 1-7　调整 Spyder 编辑器图像显示尺寸

4. 重剑无锋

对于量化分析的开发平台，笔者主张直接使用 zwPython 内置的 Spyder 开发平台。

Spyder 的工作界面，经过多年一线数据分析实盘操作人员的反馈调整和设计优化，对于数据分析工作者而言，已经是一种非常理想的工作界面，具体理由如下。

- Spyder（前身是 Pydee），是一个强大的交互式 Python 语言开发环境，提供高级的代码编辑、交互测试、调试等特性，支持包括 Windows、Linux 和 OS X 系统。
- Spyder 最早发布于 2009 年，经过多年的升级优化，目前已经非常成熟，能够在最大程度上减少各种 Bug 对于实盘操作的干扰。

- Spyder 默认界面布局，如图 1-8 所示，类似 MATLAB，集中了代码编辑，项目管理，变量检查与图形查看等多种功能，这种界面布局也是金融工程、量化分析行业的标准工作界面。

GUI 用户界面，其实类似电脑的机箱，虽然华丽，但只是表层的东西，绝非核心因素。其实许多服务器采用的 Linux 操作系统，为了追求极致的性能，还在使用传统的纯文本界面，根本没有所谓的 GUI 用户接口。

笔者在设计 zwQuant 极宽量化软件时，无论是最初的 zwQuant，还是 zQuant-core（内核版），都强调简单实用，这些案例是“kiss 法则”（保持简单）在软件工程中的具体应用。

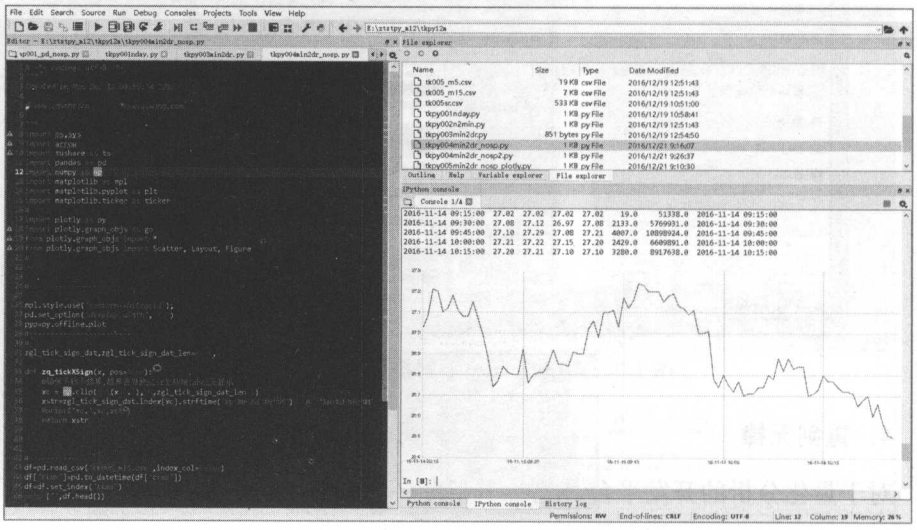


图 1-8　Spyder 工作界面

也许，大家觉得 Spyder 的界面过于朴素，这种朴素源自开源的历史与传承，大家可以看看一些著名开源项目的网站，网页都非常简单朴素，有些甚至还是互联网起步阶段的文本模式。

- http://www.apache.org：apach 开源项目网站。
- https://github.com：GITHUB 开源项目网站。
- http://www.lfd.uci.edu/～gohlke/Pythonlibs：LFD 二进制 Python 模块库。
- http://mirrors.163.com/：网易开源镜像网站。

- http://mirrors.sohu.com/：搜狐开源镜像网站。

幸运的是，如今很多成功的互联网企业，如谷歌、百度也继承了这种朴素简练的传统，搜索引擎的首页都是大片的空白，类似中国传统书法的“留空”，只有简简单单的搜索框。

在这种朴素的背后是一种“重剑无锋”的体现。

目前，Python 语言已经是数据分析、人工智能、编程教育的行业标准编程语言，大家无须争议。

在大家试图质疑这些问题的时候，请好好重新审视一下软件工程的名言：

“Don't Reinvent the Wheel”——不要重复发明轮子。

1.5　Python 命令行模式

Python 命令行模式与普通的命令行模式不同，因为集成了 Python 的运行环境参数。

许多新用户都是直接使用 Windows 软件内置的 Dos 命令，进入 Dos 命令行，直接运行 pip 命令，这样会出错，因为没有绑定 Python 运行环境。

正确的方法是，运行 Python 目录下的 WinPython Command Prompt.exe 程序，如图 1-9 所示。

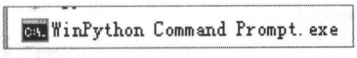


图 1-9　WinPython Command Prompt.exe 程序

- Python 27 版本，py27\WinPython Command Prompt.exe。
- Python 35 版本，py35\WinPython Command Prompt.exe。

运行后，会自动进入 Python 对应的子目录。

- Python 27 版本的目录是：x:\zwPython\py27\Python-2.7.10.amd64\。
- Python 35 版本进行了优化，目录是：x:\zwPython\py35\scripts\。

1.6 Notebook 模式

zwPython 内置的 Notebook 支持模式，目前已经是 Python 源码交流的常用模式，事实上，Notebook 已经是数据分析信息分享的 Web 标准模式。

Notebook 模式文件的后缀名是.ipynb，类似 IE 的 MHT 网页打包格式，支持文字格式、排版、图像。运行方法如下：

- 进入 Python 35 目录
- 单击运行 Jupyter Notebook.exe 程序

Jupyter Notebook.exe 程序类似单机的本地 Web 服务器软件。

如图 1-10 所示，程序运行后会自动调用默认浏览器，并访问默认网址：http://localhost:8888/tree。

“.ipynb”格式文件使用方法如下。

- 运行 Jupyter Notebook.exe 程序，进入 Notebook 模式。
- 单击右上角的“Upload”按钮，或者用鼠标直接拖放“.ipynb”格式文件到浏览器窗口。

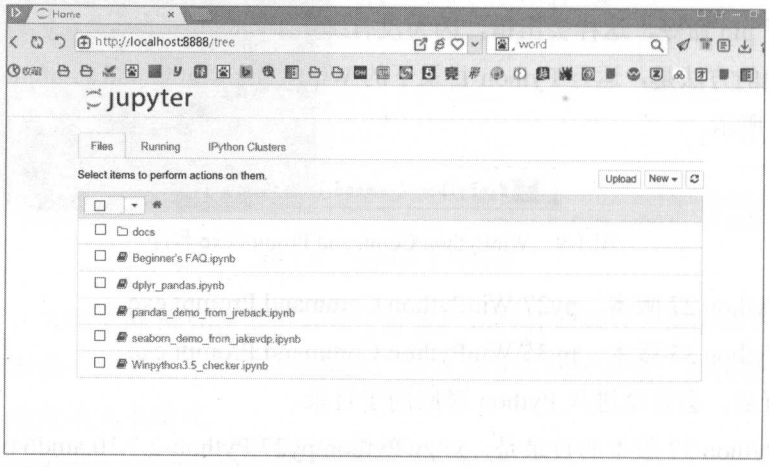


图 1-10 Notebook 模式

- 再单击文件名右侧的“Upload”按钮即可上传文件。
- 上传文件后，单击相应的文件名，即可看到相应的脚本内容，以及运行结果和图片。

具体效果如图 1-11 所示，但效果图会根据文件内容不同而有所不同。

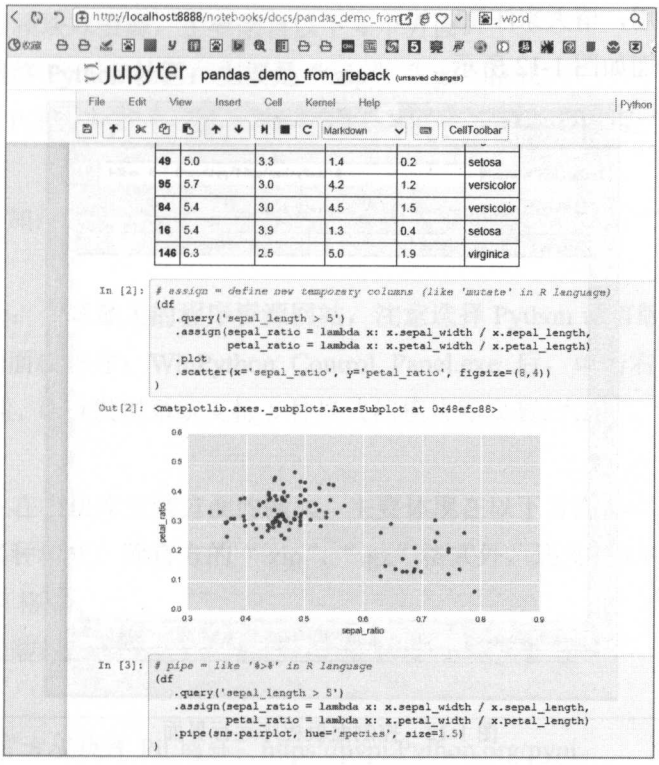


图 1-11　Notebook 运行模式

1.7　模块库控制面板

Python 的强大和方便除体现在海量的内置模块上，还体现在绿色、灵活的模块库管理功能上。

一些简单的模块或 Python 函数，可以直接复制到目录：

```
py64\python-2.7.9.amd64\Lib
```

1. 模块库更新与增删

zwPython 的模块库管理直接使用 WinPython 的控制面板程序：WinPython Control Panel.exe。

控制面板程序 WinPython Control Panel.exe 位于 Py 35 目录下，不同版本位置不同，不能混用，请大家注意。

运行后界面如图 1-12 所示。

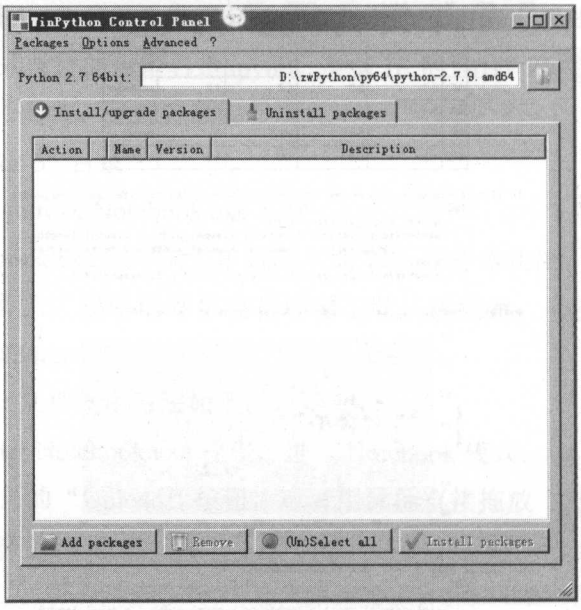


图 1-12　控制面板程序运行界面

2．zwPython 模块库的安装流程

zwPython 模块库的安装流程如下。

- 把下载的 Python 模块库复制到任意一个目录。
- 单击左下角的“Add packages”按钮，从模块目录选择模块文件名即可完成模块库的添加。一次可选择添加多个模块库文件，如果模块库版本不对，则会出现提示对话框显示相关的出错模块名称；zwPython 系统是基于 64 位 Python 3.x 版本，因此下载模块，请选择对应的版本。
- 添加完毕后，单击右下方的“Install packages”即可完成模块库的安装。

需要注意的是：

- 模块安装完成后可以删除相关的模块文件，不影响程序使用。
- 多个模块安装时，每次最好不要超过 20 个，以免出错。

3. 模块库资源

zwPython 模块库资源，主要来自以下 4 个方面。

- 各大网络 Python 社区：主要是“.zip”、“.gz”格式。
- PyPI（Python Package Index）：Python 官方模块库，主要是“.zip”、“.gz”格式。
- LFD：加州欧文大学的非官方 Python 集成模块库，主要是“.exe”、“.whl”格式。
- GitHub：全球最大的程序资源网站，注意选择 Python 语言版本。

运行控制面板程序：WinPython Control Panel.exe 后，单击右下角的“Add packages”按钮，可以发现系统支持多种格式的模块库安装：“.zip”、“.gz”、“.exe”、“.whl”。

zwPython 在模块库安装方面的强大，主要体现在以下方面。

- 支持多种格式：除官方的“.zip”、“.gz”格式外，还支持 LFD 的“.exe”、“.whl”格式。
- 绿色安装：一次安装，随处运行，支持 U 主便携式开发。

> Python 官方模块库 Pil 网址：https://pypi.Python.org/pypi。
>
> Github 网址：https://github.org。
>
> LFD 非官方模块资源网址：http://www.lfd.uci.edu/~gohlke/Pythonlibs/。
>
> （LFD 采用集成方式打包，特别适用于 OpenCV、CUDA 等大型模块库安装）
>
> LFD 全称是：Laboratory for Fluorescence Dynamics，University of California，Irvine.，动力学实验室，加利福尼亚大学/加州欧文大学。
>
> 加州欧文大学（简称为 UCI 或 UC Irvine，又常被译作加州大学欧文分校）成立于 1965 年，是加州大学 10 个校区之一，位于美国加州。

4. 模块库维护更新

运行控制面板程序：WinPython Control Panel.exe，还提供了模块库的维护和升级功能，如图 1-13 所示，单击菜单：Options→Repair packages。

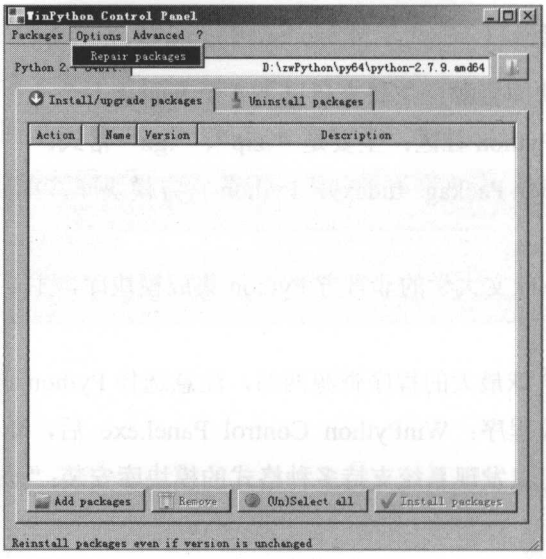


图 1-13　模块库维护

5. 系统关联

如图 1-14 所示，运行控制面板程序：WinPython Control Panel.exe，还提供系统关联功能（通常无须采用关联模式）。

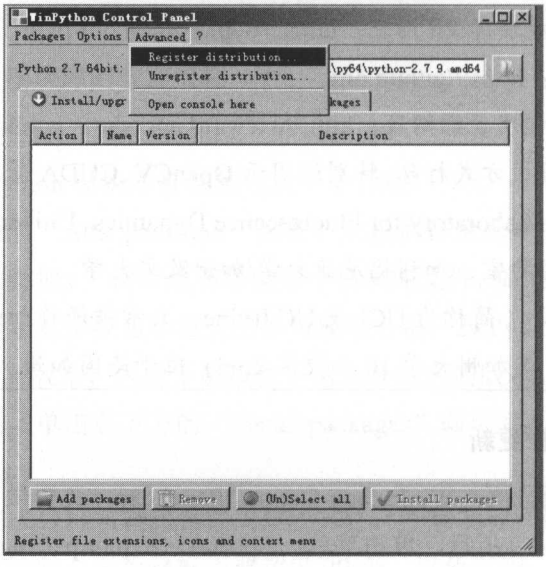


图 1-14　系统关联

- 单击菜单“Advanced→Register distribution”，即可将 zwPython 关联到 Windows 系统，关联后，可以直接在资源浏览器运行“.py”脚本文件，另外，还可以增加鼠标右键的“.py”脚本文件与“spyder”IDE 程序的关联编辑功能。
- 单击菜单“Advanced→Unregister distribution”，即可解除关联。

通常，无须采用关联模式。

1.8　使用 pip 更新模块库

有时，由于各种原因，使用控制台安装模块库会出现失败现象，或者需要批量更新模块库，这时，可以使用 pip 模块管理程序。

1. pip 常用命令

（1）列出已安装的包

```
pip freeze or pip list
```

（2）导出 requirements.txt

```
pip freeze ><目录>/requirements.txt
```

（3）在线安装：安装包、模块库

```
pip install <包名>或 pip install -r requirements.txt
```

（4）指定版本

通过使用==、>=、<=、>、<等符号来指定版本，不写则安装最新版本。

requirements.txt 内容格式为：

```
APScheduler==2.1.2
Django==1.5.4
MySQL-Connector-Python==2.0.1
MySQL-python==1.2.3
PIL==1.1.7
South==1.0.2
django-grappelli==2.6.3
django-pagination==1.0.7
```

（5）安装本地安装包

```
pip install <目录>/<文件名>或 pip install --use-wheel --no-index
```

```
--find-links=wheelhouse/ <包名>
```

注意，<包名>前有空格。

可简写为：

```
pip install --no-index -f=<目录>/ <包名>
```

（6）卸载包

```
pip uninstall <包名>或 pip uninstall -r requirements.txt
```

（7）升级包

```
pip install -U <包名>
```

（8）升级 pip

```
pip install -U pip
```

（9）显示包所在的目录

```
pip show -f <包名>
```

（10）搜索包

```
pip search <搜索关键字>
```

（11）查询可升级的包

```
pip list -o
```

（12）下载包而不安装

```
pip install <包名> -d <目录>或 pip install -d <目录> -r requirements.txt
```

（13）打包

```
pip wheel <包名>
```

（14）国内 pypi 镜像

```
pypi.v2ex.com/simple    V2EX
http://pypi.douban.com/simple   豆瓣
http://mirrors.aliyun.com/pypi/simple/ 阿里云（推荐使用）
http://pypi.mirrors.ustc.edu.cn/simple/ 中国科学技术大学
https://pypi.tuna.tsinghua.edu.cn/simple        清华大学
http://pypi.hustunique.com/  华中理工大学
http://pypi.sdutlinux.org/  山东理工大学
http://pypi.mirrors.ustc.edu.cn/  中国科学技术大学
http://mirrors.sohu.com/python/ 搜狐镜像
```

（15）指定镜像安装源

```
pip install <包名> -i http://pypi.v2ex.com/simple
```

其他更多有关 pip 的使用细节，大家可以自行搜索。

2. pip 安装模版

为了方便大家使用 pip 安装新的模块库，zwPython 集成了一个 pip01.bat 批命令模版，位于相关的目录下。

pip01.bat 批命令内容如下：

```
rem tushare
pip install --upgrade tushare -i https://pypi.tuna.tsinghua.edu.cn/simple
```

其中，tushare 是示例的模块库名称，请大家自行改为需要安装更新的模块库名称。

这个 pip01.bat 批命令会自动更新指定的模块库，如果找不到对应的模块，则会重新安装。

因为 Python 官网速度很慢，所以，我们在 pip01.bat 批命令中使用了国内的镜像源，如果出现网络问题，大家可以根据前面介绍的 PYPI 镜像站点或者自行搜索，更换对应的镜像网站即可。

3. pip 参数解释

pip 部分参数及其含义如表 1-1 所示。

表 1-1　pip 部分参数及其解释

参　数	参数含义
install	安装包
uninstall	卸载包
freeze	按着一定格式输出已安装包列表
list	列出已安装包
show	显示包详细信息
search	搜索包，类似于 yum 里的 search
wheel	按 requirements.创建模块包
zip	不推荐，Zip individual packages
unzip	不推荐，Unzip individual packages
bundle	不推荐，Create pybundles
download	下载模块
hash	计算模块包的 Hash 哈希数值
help	当前帮助

续表

参　数	参数含义
-h, --help	显示帮助
-v, --verbose	更多的输出，最多可以使用 3 次
-V, --version	显示版本信息，然后退出
-q, --quiet	最少的输出
--log-file <path>	覆盖记录 verbose 错误日志，默认文件/root/.pip/pip.log
--log <path>	不覆盖记录 verbose 输出的日志
--retries <retries>	重试次数（默认 5 次）
--trusted-host <hostname>	可信任站点，不包括 https 站点
--timeout <sec>	连接超时时间（默认 15 秒）
--cert <path>	证书
--cache-dir <dir> cache	目录
--isolated	绝对模式，无视 Python 环境和用户设置
--upgrade	如果已安装就升级到最新版

4．pip-install 参数选项

install 是最常用的 pip 参数，install 参数选项及其含义如表 1-2 所示。

表 1-2　install 参数选项及其含义

参　数	参数含义
-c, --constraint <file>	约束，使用给定的约束限制版本文件，此选项可多次使用
-e, --editable <path/url>	可编辑的<路径/网址>在“开发模式”下安装一个项目
-r, --requirement <file>	要求<文件>，按给定要求文件安装模块
-b, --build <dir>	建造模块包
-t, --target <dir>	目标<目录>
-d, --download <dir>	下载到<目录>
--src <dir> SRC	目录查看编辑项目
-U, --upgrade	所有的包升级到最新
--force-reinstall	升级时强制重新安装，即使它们已经是最新的了
-I, --ignore-installed	忽略已经安装的模块包（与 Reinstalling 相反）
--no-deps	不安装依赖包
--install-option <options>	安装选项<选项>，使用 setup.py 的额外参数
--global-option <options>	全局选项
--egg	采用 eggs 模式安装，不用默认的“flat”模式
--root <dir>	根目录，安装使用的根目录

续表

参　数	参数含义
--prefix <dir>	安装前缀目录
--compile	编译 Py 文件为 Pyc 代码
--no-compile	不编译 Py 文件为 Pyc 代码
--no-use-wheel	不使用 Wheel 模块包
--no-binary <format_control>	不使用二进制模块包
--only-binary <format_control>	不使用源码模块包，只用二进制模块包
--pre	包括预处理和开发版本
--no-clean	不清除 Build 创建目录
--require-hashes	使用 Hash 验证

第 2 章　Python 入门案例

考虑到很多刚入门的读者对于 Python 不甚了解，所以在此特意增加了一章 Python 入门案例套餐，通过几个简单的 Python 入门程序，帮助大家尽快掌握 Python 语言，熟悉开发环境。

本书采用的是：逆向式 Python 语言教学模式，先通过解压即用的 zwPython 开发平台和入门案例套餐，让大家对于 Python 语言有个基本的感性认识，再开始介绍 Python 语言的基本语法。

2.1　案例 2-1：第一次编程“hello,ziwang”

软件安装完毕，我们就可以开始编写、运行 Python 脚本程序了。

- 单击工具栏的“读取”按钮，打开“py_demo\”目录下的 py301.py 脚本文件。
- 单击工具栏中部的绿色“▸”运行按钮。

程序很简单，只有一行代码：

```
print("hello,ziwang.com")
```

如图 2-1 所示，运行后，在右下角的输出窗口可以看到“hello,ziwang.com”的字样，表示运行成功。注意，输出面板是 IPython Console。

大家可以自己修改引号里面的文字，看看输出效果，注意，此处必须是英文字符和标点，中文字符的处理我们后面再讲解。

1. 简单调试

下面，我们学习最简单的调试，如图 2-2 所示，去掉代码左边的引号，再单击“▶”运行按钮。

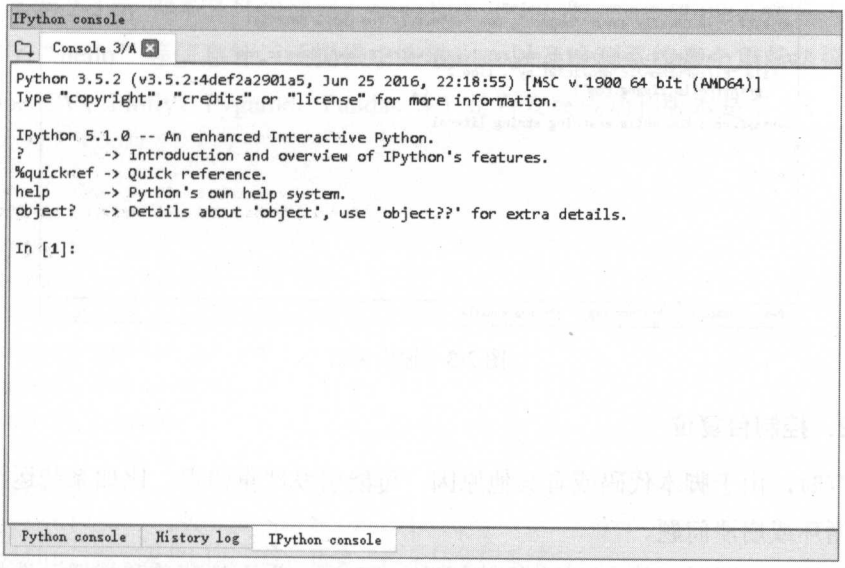


图 2-1　输出面板

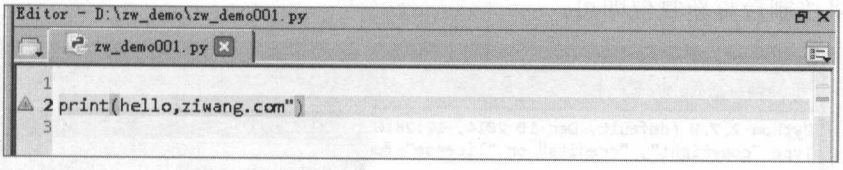


图 2-2　修改代码

右下角的输出窗口如图 2-3 所示。

图 2-3 显示输出有错误，注意这行文字：

```
File "e:/zwPython/py_demo/zc201.py", line 2
```

其中的“line 2”表示出错的代码位于第二行。

出错信息是：

```
SyntaxError: EOL while scanning string literal
```

表示是字符串应用错误，我们加上引号即可。

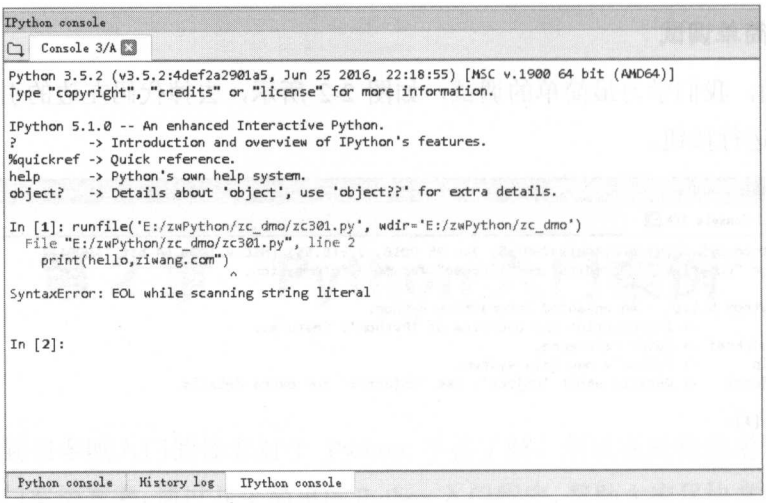


图 2-3　输出窗口

2. 控制台复位

有时，由于脚本代码或者其他原因，可能引发严重错误，比如系统运行时出现死循环或崩溃问题。

如图 2-4 所示，单击 IDE 右侧中部的“Restart”下拉菜单和按钮，选择相应选项将控制台重新复位即可。

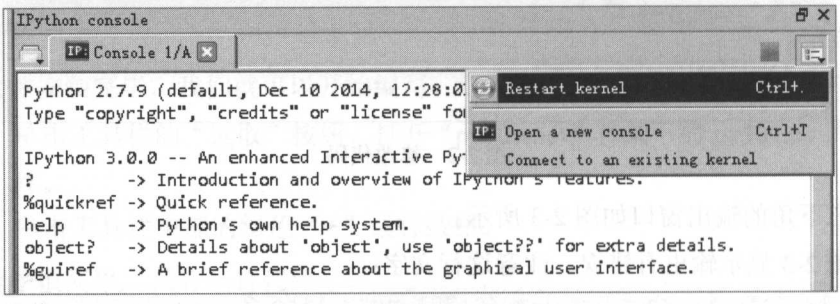


图 2-4　系统复位

2.2　案例 2-2：增强版“hello,ziwang”

下面我们运行一个增强版的“hello,ziwang”。

- 单击工具栏的“读取”按钮，打开“py_demo\”目录下的 py302.py 脚本文件。
- 单击工具栏中部的绿色“▸”运行按钮。

案例 2-2.py 脚本文件很简单，核心程序才十几行，不过功能非常强大，除输出文字“hello”等信息外，还提供中文输出，以及检测系统多个重量级模块（比如 OpenCV、Plotly、Pygame、Pandas 等）是否安装成功和版本是多少。

```
# -*- coding: utf-8 -*-

import sys,os,re
import cv2
import arrow,plotly

import pandas as pd
import tushare as ts
import pygame

print("hello,zwPython 2017")
print("hello,TopQuant,TopFootball")
print("极宽量化回溯系统，极宽足彩量化分析系统")
print("")
print("python ver:",sys.version)
print("")
print("re ver:",re.__version__)
print("arrow:",arrow.__version__)
print("plotly:",plotly.__version__)
print("")
print("pandas ver:",pd.__version__)
print("tushare ver:",ts.__version__)
print("")
print("pygame ver:",pygame.ver)
print("opencv ver:",cv2.__version__)
```

如果案例 2-2 运行无误，输出面板结果如下（不同版本的细节略有差别）：

```
hello,zwPython 2017
hello,TopQuant,TopFootball
极宽量化回溯系统，极宽足彩量化分析系统

python ver: 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC
```

```
v.1900 64 bit (AMD64)]

   re ver: 2.2.1
   arrow: 0.10.0
   plotly: 2.0.0

   pandas ver: 0.19.1
   tushare ver: 0.6.2

   pygame ver: 1.9.2b1
   opencv ver: 3.1.0
```

案例 2-2 表明，即使是初学者，采用 Basic 的过程模式编写简单代码，也能完成很复杂的功能。

2.3 案例 2-3：列举系统模块库清单

案例 2-3 文件名是 py_demo\py303mlst.py，代码很简单，如下所示：

```
# -*- coding: utf-8 -*-

import numpy as np
import scipy as sp
import pandas as pd
import pip

# =======================
x10=pip.get_installed_distributions();
df=pd.DataFrame();
df['name']=x10
print(df.head())

df.to_csv('tmp/m10.csv',index=False)
```

运行结果如下：

```
                    name
0              3to2 1.1.1
1      zope.interface 4.3.2
```

```
2  zipline-cn-databundle 0.4
3               zict 0.1.0
4               zarr 2.1.3
```

运行后，会在 tmp 目录生成一个名称为 m10.csv 的数据文件，文件里面是当前 zwPython 所安装的第三方模块库名称和版本号。在《zwPython 用户手册》中，《zwPy3.5 内置模块库列表》就是这样完成的。

案例 2-3 代码虽然简单，但调用了 Pandas（潘达思）数据分析模块，以及很少在程序中直接调用的 pip 模块库，来获取内置模块清单。

案例 2-3 是笔者在编程中实际使用的脚本程序，大家可以自行保存，用于日常查看自己的 Python 运行环境的模块库安装情况。

2.4　案例 2-4：常用绘图风格

可视化计算是数据分析的重要组成部分，Python 可视化计算、数据分析和量化分析最重要的模块是：Matplotlib。

如今，虽然新一代绘图模块 Plotly 横空出世，且在互动性方面具有压倒性的优势，但由于历史原因，Pandas（潘达思）数据分析模块和很多第三方内置的绘图模块还是使用 Matplotlib。绘制一般的简单图形，由此可见还是使用 Matplotlib 比较方便。

老版本的 Matplotlib 默认的绘图风格与现代设计风格有些差距，因此，需要增加一些扩展风格包。2017 年 1 月，期待已久的 Matplotlib 2.0 终于发布，在风格方面有所优化。

通常，Matplotlib 的绘图风格只有几种：

```
'bmh','dark_background','fivethirtyeight','ggplot','grayscale','
default'。
```

zwPython 将 Matplotlib 的绘图风格增加到 20 多种：

```
'dark_background', 'seaborn-colorblind', 'classic', 'grayscale',
'seaborn-dark-palette',
'seaborn-ticks','fivethirtyeight','seaborn-paper',
'seaborn-poster', 'seaborn-white',
'seaborn-muted','seaborn-notebook','seaborn-dark','bmh',
```

```
'ggplot',
    'seaborn-darkgrid','seaborn-whitegrid','seaborn-bright',
'seaborn-pastel', 'seaborn-talk','seaborn-deep'
```

以上风格，除 Matplotlib 调用外，也可直接应用到 Pandas 绘图语句中，需要注意的是，使用不同的版本、处于不同的 Python 运行环境时，以上具体参数在细节方面会有所差异。

案例 2-4 文件名为 py304dr.py，代码使用 Matplotlib 绘图指令进行绘图，全部代码如下：

```
    # -*- coding: utf-8 -*-

    import numpy as np
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import pandas as pd

    def dr_xtyp(_dat):
        #xtyp=['bmh','dark_background','fivethirtyeight','ggplot',
'grayscale','default'];
        for xss in plt.style.available:
            plt.style.use(xss);print(xss)
            plt.plot(_dat['Open'])
            plt.plot(_dat['Close'])
            plt.plot(_dat['High'])
            plt.plot(_dat['Low'])
            fss="tmp\\stk001_"+xss+".png";plt.savefig(fss);
            plt.show()

    # ======================
    df = pd.read_csv('dat\\appl2014.csv',index_col=0,parse_dates=[0],
encoding='gbk')
    d30=df[:30];
    dr_xtyp(d30);
```

如图 2-5 所示，运行后会在 tmp 目录下生产一系列各种不同风格的图形，可以保存一下，作为日后编程绘图的参考。

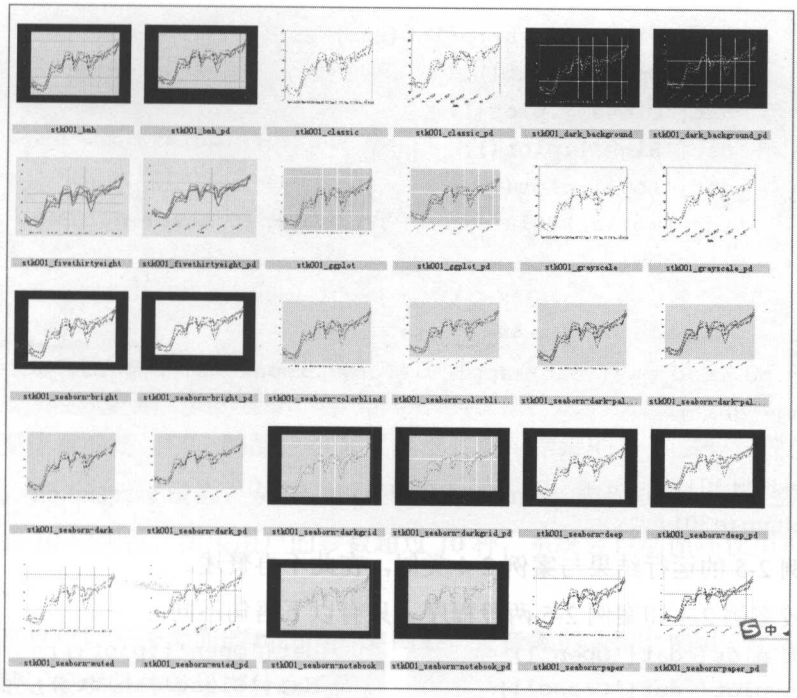


图 2-5　不过风格的输出图形

2.5　案例 2-5：Pandas 常用绘图风格

案例 2-5 文件名为 py305drpd.py，使用 Pandas（潘达思）数据分析模块内置的 plot 命令绘制图形，代码如下：

```
    # -*- coding: utf-8 -*-

    import numpy as np
    import matplotlib as mpl
    import matplotlib.pyplot as plt
    import pandas as pd

    def dr_xtyp(_dat):
        #xtyp=['bmh','dark_background','fivethirtyeight','ggplot',
'grayscale','default'];
        for xss in plt.style.available:
```

```
        plt.style.use(xss);print(xss)
        _dat['Open'].plot();
        _dat['Close'].plot();
        _dat['High'].plot();
        _dat['Low'].plot();
        fss="tmp\\stk001_"+xss+"_pd.png";plt.savefig(fss);
        plt.show()

# ======================
df = pd.read_csv('dat\\appl2014.csv',index_col=0,parse_dates=[0],
encoding='gbk')

d30=df[:30];
dr_xtyp(d30);
```

案例 2-5 的运行结果与案例 2-4 类似，在此不再赘述。

比较案例 2-4 和案例 2-5 两段程序，只有以下语句不同：

```
plt.plot(_dat['Open'])
plt.plot(_dat['Close'])
plt.plot(_dat['High'])
plt.plot(_dat['Low'])
```

```
_dat['Open'].plot();
_dat['Close'].plot();
_dat['High'].plot();
_dat['Low'].plot();
```

运行后截图，除 Pandas 优化的细节外，两者风格大体相同。具体细节，请读者自己参看 tmp 目录下的相关图片。

Pandas（潘达思）数据分析模块是最重要的数据分析工具，大家必须了解其基本功能和常用的函数命令。

所以，笔者特意增加了本案例，强调 Padnas（潘达思）数据分析软件的使用，如果还有读者对于本案例有困惑，请补习一下 Padnas（潘达思）基础知识，再继续深入学习。

2.6 案例 2-6：常用颜色表 cors

下面我们再介绍一些常用的颜色组合，在极宽 zsys 模块中内置了多种常用的颜色组合，以下是 zsys 模块颜色组合设置的部分相关代码：

```
import matplotlib.colors
from matplotlib import cm

cors_brg=cm.brg(np.linspace(0,1,10))
cors_hot=cm.hot(np.linspace(0,1,10))
cors_hsv=cm.hsv(np.linspace(0,1,10))
cors_jet=cm.jet(np.linspace(0,1,10))
cors_prism=cm.prism(np.linspace(0,1,10))
cors_Dark2=cm.Dark2(np.linspace(0,1,10))
cors_Vega10=cm.Vega10(np.linspace(0,1,10))
cors_Vega20=cm.Vega20(np.linspace(0,1,10))
```

需要注意的是，以上的颜色组合代码没有使用常规的 rgb 参数设置模式，而是使用的 np.linspace 函数，从 Matplotlib 的 Corlormap 中提取 10 个颜色，组成一个系列的 10 种颜色，如果用户的参数超过 10 种，系统会自动循环使用每组的 10 个颜色。

大家也可以参考上面的 np.linspace 函数，并在案例 2-6 中使用 20、30、50 等其他参数，看看具体效果有什么不同。

案例 2-6 文件名为 py306_cors.py，说明如何绘制 Matplotlib 内置的颜色图谱，以及极宽量化模块内置的颜色组合。

颜色是图表设计的一个核心部分，事实上，现代图表与传统图表的一个重要差异就是颜色的组合，Matplotlib 2.0 升级版本，其中最重要的升级之一就是颜色体系更加适合现代的设计风格。

案例 2-6 内置了以下两个函数

- dr_cormap()函数：绘制 Matplotlib 内置的颜色图谱，不同的 Python 环境，因 Matplotlib 版本和辅助模块的差异，具体数目会有所不同。
- dr_cors_sys()函数：绘制极宽量化模块内置的颜色组合，一共有 8 种。

dr_cormap()函数代码如下：

```
def dr_cormap(fcor='dat/cormap.dat'):
    #font = FontProperties(fname=r"c:\windows\fonts\simsun.ttc",
size=14)
    clst=zt.f_lstRdTxt(fcor);
    ds=pd.Series(range(5,25));
    for xc,cor in enumerate(clst):
```

```
        css=cor[0]
        xss='cm.'+str(css)+'(np.linspace(0,1,10))'
        print(xc,'#',css,xss)
        cor2=eval(xss)
        #print(css,xss,cor2)
        ds.plot(kind='bar',rot=0,color=cor2)
        plt.savefig('tmp/cm_'+css+'.png')
```

程序运行后，会在 tmp 目录下生成数十个不同的颜色组合图形文件，大家可以选择自己喜欢的图形，参考极宽的颜色设置代码，添加自己喜欢的颜色组合。

案例 2-6 也是笔者在工作中实际使用的代码程序，大家可以浏览 dat/cormap.dat 文件，查看相关的颜色参数数据，保存好输出的图形文件，特别是自己喜欢的一些颜色效果，以供日后编程参考。

第 3 章　Python 基本语法

Python 是一种学习简单、功能强大的工业级编程语言，也是一种真正的终身编程语言，适合 8～80 岁的用户学习编程，是小学生和博士生通用的编程语言。

语法是编程语言中基础的基础，即使你是一位有经验的程序员，再看看这些 Python 编程语法，也会对于相关的语法细节有更多的认识。

3.1　数据类型

Python 有 5 种基本数据类型：

- Numbers（数字）
- String（字符串）
- List（列表）
- Tuple（元组）
- Dictionary（字典）

> 注意：
>
> （1）Python 的数据类型和 C 语言的不同，有复数形式，比如(-6+4j)和(5.3-7.6j)。
>
> （2）Python 没有 char 单字符类型。

数字类型用于存储数值。

当给一个变量赋值时，Number 数据类型就会被创建：

```
x=1
y=911
```

Python 支持 4 种不同的数字类型：

- Int（有符号整型）
- Long（长整型，也可以代表八进制和十六进制）
- Float（浮点型）
- Complex（复数）

Python 常用的计算符号有：

- +，加法。
- -，减法。
- *，乘法。
- /，除法。
- //，整除。
- %，取模，余数。
- **，乘方。

案例 3-1：基本运算

案例 3-1 文件名为 py401math.py，主要介绍 Python 数值的基本运算，核心代码如下：

```
#1
print('\n#1')
x=10
y=22
z=35
print('x,y,z,',x,y,z)

#2
print('\n#2')
a=x+y;print('a=x+y,',a)
b=x-y;print('b=x-y,',b)
c=z-x*y;print('c=z-x*y,',c)

#3
print('\n#3')
```

```
a=z/x;print('a=z/x,',a)
b=z//x;print('b=z//x,',b)
c=z%x;print('c=z%x,',c)

#4
print('\n#4')
a=x**2;print('a=x**2,',a)
b=x**3;print('b=x**3,',b)
```

对应的输出信息如下：

```
#1
x,y,z, 10 22 35

#2
a=x+y, 32
b=x-y, -12
c=z-x*y, -185

#3
a=z/x, 3.5
b=z//x, 3
c=z%x, 5

#4
a=x**2, 100
b=x**3, 1000
```

3.2　字符串

字符串 String 是由数字、字母、下画线组成的一串字符，一般采用单引号或者双引号形式：

```
str='abcd'
str="hello ziwang.com"
```

Python 语言的字符串类似传统语言的字符数组模式，也可以看作字符列表，有两种取值顺序：

- 从左到右索引默认 0 开始的，最大范围是字符串长度少 1。
- 从右到左索引默认-1 开始的，最大范围是字符串开头。

如果要实现从字符串中获取一段子字符串，使用变量[头下标:尾下标]，就可以截取相应的字符串，其中下标从 0 开始算起，可以是正数或负数，下标可以为空，表示取到头或尾。

案例 3-2：字符串入门

案例 3-2 文件名为 py402str.py，主要介绍字符串的基本用法，核心代码如下：

```
dss='hello ziwang.com'
print('dss',dss)

#1
print('\n#1')
s2=dss[1:];print('s2,',s2)
s3=dss[1:3];print('s3,',s3)
s4=dss[:3];print('s4,',s4)

#2
print('\n#2')
s2=dss[-1];print('s2,',s2)
s3=dss[1:-2];print('s3,',s3)
dn=len(dss);print('dn,',dn)

#3
print('\n#3')
print('s2+s3,',s2+s3)
print('s3*2,',s3*2)
```

对应的输出信息如下：

```
dss hello ziwang.com

#1
s2, ello ziwang.com
s3, el
s4, hel

#2
s2, m
s3, ello ziwang.c
dn, 16
```

```
#3
s2+s3, mello ziwang.c
s3*2, ello ziwang.cello ziwang.c
```

在字符串运算中，加号（+）是字符串连接运算符，乘号（*）是重复操作。

案例 3-3：字符串常用方法

Python 语言的字符串其实是一种对象，内置了大量实用的字符串函数和方法，几乎包括了所有常用的 Python 字符串操作，如字符串的替换、删除、截取、复制、连接、比较、查找、分割等。

案例 3-3 文件名为 py403str2.py，主要介绍字符串内置函数和方法，所以，程序代码较长，我们分组进行说明。

程序代码	对应的输出信息
#1 dss='　hello ziwang.com,,' print('\n#1,去空格及特殊符号') s2=dss.strip().lstrip().rstrip(',') print('s2,',s2)	#1,去空格及特殊符号 s2, hello ziwang.com
#2 print('\n#2,字符串连接') s2=dss.join(['a','.','c']) print('s2,',s2) s3='s3' s3+='xx' print('s2,',s3)	#2,字符串连接 s2, a　hello ziwang.com,,.　hello ziwang.com,,c s2, s3xx
#3 print('\n#3,查找字符') css='abc1c2c3' pi=css.find('c') print('pi,',pi)	#3,查找字符 pi, 2
#4,字符串比较 s1>s2 False s1==s2 False s1<s2 True	#4,字符串比较 s1>s2 False s1==s2 False s1<s2 True

#5 print('\n#5,字符串长度') s1,s2='abc','c123' print('len(s1),',len(s1)) print('len(s2),',len(s2))	#5,字符串长度 len(s1), 3 len(s2), 4
#6 print('\n#6,大小写转换') s1,s2='abc','ABC123efg' print('大写，s1.upper(),',s1.upper()) print('小写，s2.lower(),',s2.lower()) print('大小写互换 ,s2.swapcase(),',s2.swapcase()) print('首字母大写 ,s1.capitalize(),',s1.capitalize())	#6,大小写转换 大写，s1.upper(), ABC 小写，s2.lower(), abc123efg 大小写互换，s2.swapcase(), abc123EFG 首字母大写，s1.capitalize(), Abc
#7 print('\n#7,分割字符串') s2=' hello, ziwang,com,,' print('s2.split,',s2.split(','))	#7,分割字符串 s2.split, [' hello', ' ziwang', 'com', '', '']

3.3 List 列表

List 列表，使用[ ]标识，是 Python 中最常用的数据类型，List 列表类似传统语言中的数组，但更加灵活强大。

List 列表支持字符、数字、字符串，甚至可以包含列表（所谓嵌套）。

列表中的每一个元素都分配一个数字，即它的位置或索引，第一个索引是 0，第二个索引是 1，依此类推。从右到左索引默认从 1 开始，下标可以为空，表示取到头或尾。

案例 3-4：列表操作

案例 3-4 文件名为 py404list.py，主要介绍列表的基本用法，核心代码如下：

```
#1
print('\n#1')
zlst=['hello','ziwang','.','com']
vlst=['Top','Quant','.','vip']
print('zlst,',zlst)
print('vlst,',vlst)
```

```
#2
print('\n#2')
s2=zlst[1:];print('s2,',s2)
s3=zlst[1:3];print('s3,',s3)
s4=vlst[:3];print('s4,',s4)

#3
print('\n#3')
print('s2+s3,',s2+s3)
print('s3*2,',s3*2)
```

对应的输出信息如下：

```
#1
zlst, ['hello', 'ziwang', '.', 'com']
vlst, ['Top', 'Quant', '.', 'vip']

#2
s2, ['ziwang', '.', 'com']
s3, ['ziwang', '.']
s4, ['Top', 'Quant', '.']

#3
s2+s3, ['ziwang', '.', 'com', 'ziwang', '.']
s3*2, ['ziwang', '.', 'ziwang', '.']
```

加号（+）是列表连接运算符，乘号（*）是重复操作。

列表操作常用函数和方法

（1）List 列表操作包含以下函数。

- cmp(list1, list2)：比较两个列表的元素。
- len(list)：列表元素个数。
- max(list)：返回列表元素最大值。
- min(list)：返回列表元素最小值。
- list(seq)：将元组转换为列表。

（2）List 列表操作包含以下方法。

- list.append(obj)：在列表末尾添加新的对象。
- list.count(obj)：统计某个元素在列表中出现的次数。

- list.extend(seq)：在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）。
- list.index(obj)：从列表中找出某个值第一个匹配项的索引位置。
- list.insert(index, obj)：将对象插入列表。
- list.pop(obj=list[-1])：移除列表中的一个元素（默认最后一个元素），并且返回该元素的值。
- list.remove(obj)：移除列表中某个值的第一个匹配项。
- list.reverse()：反向列表中元素。
- list.sort([func])：对原列表进行排序。

3.4 Tuple 元组

Tuple 元组用“()”标识，是 List 列表数据格式的简化版本，不能二次赋值，类似只读列表。。

元组用“()”标识，内部元素用逗号隔开。但是元组不能二次赋值，类似只读列表。

案例 3-5：列表操作

案例 3-5 文件名为 py405tuple.py，主要介绍元组的基本用法，核心代码如下：

```
#1
print('\n#1')
zlst=('hello','ziwang','.','com')
vlst=('Top','Quant','.','vip')
print('zlst,',zlst)
print('vlst,',vlst)

#2
print('\n#2')
s2=zlst[1:];print('s2,',s2)
s3=zlst[1:3];print('s3,',s3)
s4=vlst[:3];print('s4,',s4)

#3
```

```
print('\n#3')
print('s2+s3,',s2+s3)
print('s3*2,',s3*2)
```

对应的输出信息如下：

```
#1
zlst, ('hello', 'ziwang', '.', 'com')
vlst, ('Top', 'Quant', '.', 'vip')

#2
s2, ('ziwang', '.', 'com')
s3, ('ziwang', '.')
s4, ('Top', 'Quant', '.')

#3
s2+s3, ('ziwang', '.', 'com', 'ziwang', '.')
s3*2, ('ziwang', '.', 'ziwang', '.')
```

3.5　Dictionary 字典

Dictionary 字典，用“{ }”标识，由索引（key）和它对应的值（value）组成，是除了 List 列表以外，Python 中最灵活的内置数据结构类型，类似其他语言的 k-v 数据类型。列表是有序的对象结合，字典是无序的对象集合。

Python 字典是一种可变容器模型，且可存储任意类型对象，如字符串、数字、元组等其他容器模型。

字典由键和对应值成对组成，也被称作关联数组或哈希表，字典中的元素通过 key 关键词来存取，而不是通过偏移存取。

案例 3-6：字典操作

案例 3-6 文件名为 py406dict.py，主要介绍字典的基本用法，核心代码如下：

```
#1
print('\n#1')
zdict={}
zdict['w1']='hello'
zdict['w2']='ziwang.com'
print('zdict,',zdict)
```

```
#2
print('\n#2')
vdict={'url1':'TopQuant.vip'
      ,'url2':'www.TopQuant.vip'
      ,'url3':'ziwang.com'}
print('vdict,',vdict)


#3
print('\n#3')
s2=zdict['w1'];print('s2,',s2)
s3=vdict['url2'];print('s3,',s3)
```

对应的输出信息如下：

```
#1
zdict, {'w2': 'ziwang.com', 'w1': 'hello'}

#2
vdict, {'url3': 'ziwang.com', 'url2': 'www.TopQuant.vip', 'url1':
'TopQuant.vip'}

#3
s2, hello
s3, www.TopQuant.vip
```

字典内置函数和方法

（1）Python 字典包含了以下内置函数。

- cmp(dict1, dict2)：比较两个字典元素。
- len(dict)：计算字典元素个数，即键的总数。
- str(dict)：输出字典可打印的字符串标识。
- type(variable)：返回输入的变量类型，如果变量是字典就返回字典类型。

（2）Python 字典包含了以下内置方法。

- radiansdict.clear()：删除字典内所有元素。
- radiansdict.copy()：返回一个字典的浅复制。
- radiansdict.fromkeys()：创建一个新字典，以序列 seq 中的元素做字典的键，

val 为字典所有键对应的初始值。

- radiansdict.get(key, default=None)：返回指定键的值，如果值不在字典中，则返回 Default 值。
- radiansdict.has_key(key)：如果键在字典 Dictionary 里，则返回 True，否则返回 False。
- radiansdict.items()：以列表返回可遍历的（键，值）元组数组。
- radiansdict.keys()：以列表返回一个字典所有的键。
- radiansdict.setdefault(key, default=None)：和 get()类似，但如果键已经不存在于字典中，将会添加键并将值设为 Default。
- radiansdict.update(dict2)：把字典 dict2 的键/值对更新到 Dictionary 里。
- radiansdict.values()：以列表返回字典中的所有值。

3.6　数据类型转换

有时候，需要对数据内置的类型进行转换，只要将数据类型作为函数名即可。

以下几个内置的函数可以执行数据类型之间的转换，这些函数返回一个新的对象，表示转换的值。

常用的类型转换函数有以下几种。

- int(x [,base])：将 x 转换为一个整数。
- long(x [,base])：将 x 转换为一个长整数。
- float(x)：将 x 转换到一个浮点数。
- complex(real)：创建一个复数。
- str(x)：将对象 x 转换为字符串。
- repr(x)：将对象 x 转换为表达式字符串。
- eval(str)：用来计算在字符串中的有效 Python 表达式，并返回一个对象。
- tuple(s)：将序列 s 转换为一个元组。
- list(s)：将序列 s 转换为一个列表。
- chr(x)：将一个整数转换为一个字符。

- unichr(x)：将一个整数转换为 Unicode 字符。
- ord(x)：将一个字符转换为它的整数值。
- hex(x)：将一个整数转换为一个十六进制字符串。
- oct(x)：将一个整数转换为一个八进制字符串。

案例 3-7：控制语句

Python 语言的控制命令和其他编程语言类似，常用的有 if...else、while、for 等语句。

案例 3-7 文件名为 py407ctrl.py，介绍 Python 常用的控制命令，由于代码较长，下面进行分组说明。

第 1 组代码，说明 if...else 语句：

```
#1
print('\n1,if')
x,y,z=10,20,5
if x>y:
    print('x>y')
else:
  print('x<y')
```

对应的输出信息：

```
1,if
x<y
```

第 2 组代码，说明 elif 语句：

```
#2
print('\n#2,elif')
x,y,z=10,20,5
if x>y:
    print('x>y')
elif x>z:
  print('x>z')
```

对应的输出信息：

```
#2,elif
x>z
```

第 3 组代码，说明 while 循环语句：

```
#3
print('\n#3,whiel')
```

```
x=3
while x>0:
    print(x)
    x-=1
```

对应的输出信息：

```
#3,whiel
3
2
1
```

第 4 组代码，说明 for 循环语句的第一种用法：

```
#4
print('\n#4,for')
xlst=['1','b','xxx']
for x in xlst:
    print(x)
```

对应的输出信息：

```
#4,for
1
b
```

第 5 组代码，说明 for 循环语句的第二种用法：

```
#5
print('\n#5,for')
for x in range(3):
    print(x)
```

对应的输出信息：

```
#5,for
0
1
2
```

在 Python 控制语句中，需要注意 for 循环语句，其采用的迭代模式与传统编程语言差异较大，其他的控制语句用法都差不多。

案例 3-8：函数定义

Python 语言没有子程序，只有自定义函数，目的是方便我们重复使用相同的一段程序。将常用的代码块定义为一个函数，以备在以后你想实现相同的操作时，只要调用函数名就可以了，而不需要重复输入所有的语句。函数的定义使用 def

命令。

案例 3-8 文件名为 py408fun.py，说明自定义函数的使用，核心代码如下：

```
def f01(a,b,c):
    print('a,b,c,',a,b,c)
    a2,b2,c2,=a+c,b*2,c*2
    #
    return a2,b2,c2
#

#1
print('\n#1')
x,y,z=f01(1,2,3)
print('x,y,z,',x,y,z)

#2
print('\n#2')
x,y,z=f01(x,y,z)
print('x,y,z,',x,y,z)
```

运行结果如下：

```
#1
a,b,c, 1 2 3
x,y,z, 4 4 6

#2
a,b,c, 4 4 6
x,y,z, 10 8 12
```

以上代码和输出信息需要注意的是：

- a、b、c 的输出信息是由自定义函数 f01 完成的。
- 调用 f01 函数时，变量 x、y、z 即是输入参数，也是输出变量。
- Python 函数支持多个返回数据。

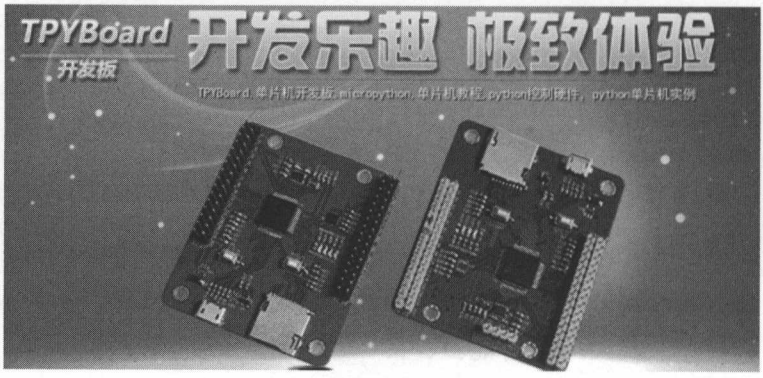
第二部分　PyBox 实战案例

PyBox 主板说明

PyBox 迷你小电脑是 PyBox 第一版，以遵照 MIT 许可的 MicroPython 为基础的硬件开发板。

PyBox，基于 STM32F405 单片机，通过 USB 接口进行数据传输。该开发板内置 4 个 LED 灯和 1 个加速度传感器，可在 3～10V 之间的电压正常工作。

PyBox 第一版，也称 TPYBoard 开发板，是由 TurnipSmart（萝卜电子）公司制作的一款 MicroPython 开发板，如下图所示。



TPYBoard 开发板

有关 TPYBoard 开发板的更多资料，请浏览网站：http://www.tpyboard.com/。

第 4 章　请让我为你点盏灯

本章，我们聪明伶俐的 Python 极客，就用自己的智慧，为大家设计一款可以通过 PyBox 极客迷你小电脑控制的智慧之灯。

案例很简单，就是通过 Python 程序，控制 PyBox 极客迷你小电脑内置的几组 LED 彩灯的打开和关闭。

零件清单

本章需要的零件如下：

（1）PyBox 极客迷你小电脑一台。

（2）Micro USB 电脑连接线一根。

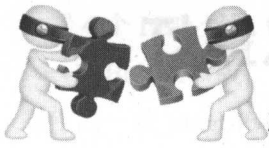
编写程序代码

本案例的程序代码只有两行，程序文件名是：TPY_001.py，该文件名表示本程序是本书第一个案例程序，全部代码如下：

```
myled = pyb.LED(4)
myled.on()
```

大家可以使用专业的 zwPython 集成平台编写代码，也可以使用普通的文本编辑器，笔者建议使用专业的 zwPython 集成平台编写代码，因为专业的集成平台不仅有语法高亮提示，而且也便于调试。

编写好代码以后，给文件命名，保存好源码文件即可。



硬件连接

本案例没有额外零件，直接使用 Micro USB 把 PyBox 迷你电脑和 PC 的 USB 接口连接即可。

有图为证

使用 Micro USB 把 PyBox 迷你电脑和 PC 的 USB 接口连接，如图 4-1 所示。

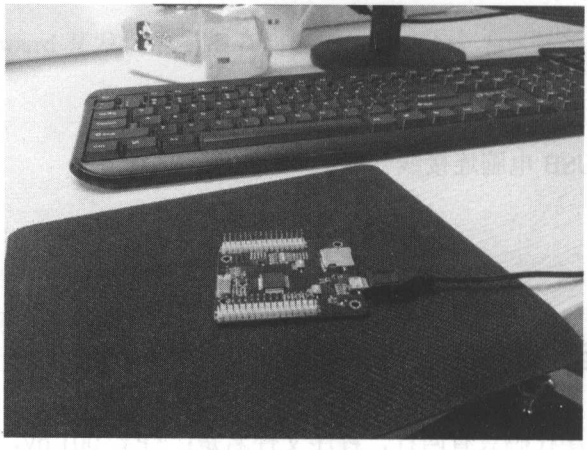


图 4-1　硬件连接图

注意事项

在连接硬件时应该注意保护好设备和元器件，在图 4-1 中，因为是临时工作台，所以我们使用了鼠标垫的反面作为工作台面板。

用鼠标垫背面作为工作台面板，是因为正面通常印有花纹，会干扰工作视线，而且鼠标垫绝缘、价格低廉，又有很好的弹性。如果是长期的固定工作台，建议购买大型的长条鼠标垫。

PyBox 是独立的硬件设备，与编写普通的 Python 程序有所不同。以上代码编写完成后，再连接 PyBox 与 PC 的 USB 接口。

正常连接成功后，PyBox 的灯会闪烁，同时，在 PC 电脑的资源管理器会出现一个新的盘符：TPFLASH，这个盘就是 PyBox。

打开 PyBox 盘，会看到上面有几个文件，其中 main.py 就是 PyBox 极客迷你小电脑，每次通电后按“Reset”（RST）复位按钮就会自动运行。

注意：这里切记要等红灯熄灭再按下“Reset”（RST）复位按钮，否则会导致板载 Flash 文件清除，以至于重新恢复出厂设置。

开始运行

完成程序代码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中并保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄灭后，再按下“Reset”（RST）复位按钮，运行 main.py 文件中的程序代码。

成功时刻

在程序中，我们点亮的是第 4 号灯，默认情况下，4 号灯是蓝色的，可以看到 PyBox 电路板上的蓝灯亮着。

进阶学习

虽然本案例点亮了 4 号灯，但是它不会熄灭，需要拔掉连接线，关闭电源才能熄灭。此外，我们拔掉 USB 连线再次连接时，蓝灯依然是点亮状态。

这说明本案例的程序代码已的确写入（类似于传统芯片的固化）了 PyBox 芯片中，而且 PyBox 启动时自动运行了程序文件 main.py。

知识点

前面通过手工方式把硬盘上的 Py01_01.py 程序文件传送到 PyBox 芯片的 main.py 文件中，其实还可以使用以下 Python 代码，自动部署程序文件到 PyBox 芯片中。注意：测试代码 Py01_01.py 放在硬盘 E:/testPython 文件夹下。PyBox 芯片组别识别为“F：盘”，读者应根据自己的计算机稍微修改一下配置信息。如图 4-2 所示。

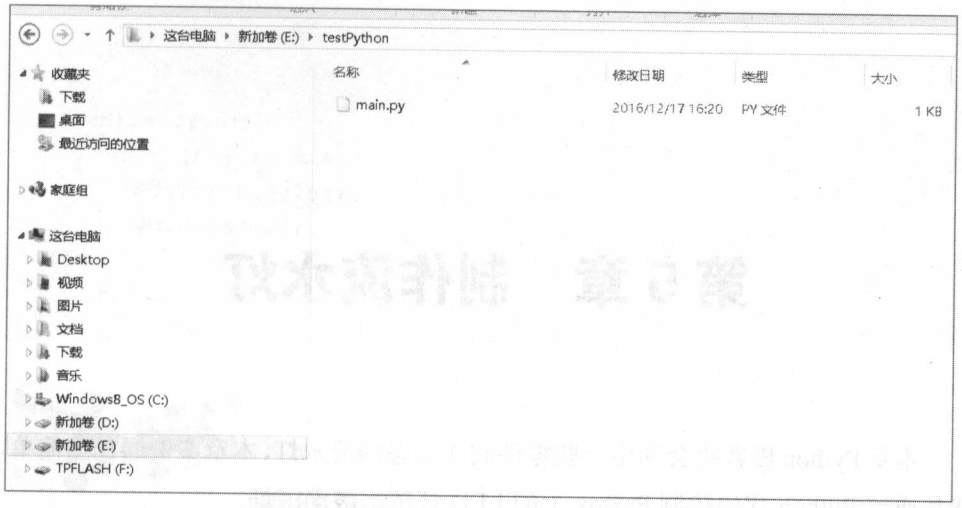


图 4-2　main.py 文件

自动部署脚本如下。

```
//硬盘上的需要调试的 Python 文件：
editMainPy = "E:/testPython/TPY_001.py"
//PyBox 上的主程序文件：
destMainPy = "F:/main.py"
import shutil
editMainPy = "E:/zwDat/k40demo/xinping/TPY_001.py"
destMainPy = "f:/main.py"
```

自动部署脚本非常简单，整个核心代码只有 5 行，很多时候需要对文件进行一些操作（比如移动、复制等），这种类型的函数被包含在 shutil(shell utility)包里，这个包里提供了很多基本操作功能，比如上面的代码我们就用到了复制文件功能，如果读者对 shutil 包感兴趣，可以去 Python 官网（https://docs.python.org/3/）查看官方文档，那里有更多的功能介绍。

自动部署环节对于初学者来说有些难度，如果暂时无法理解，可以先跳过去，等看完全书再重新学习；本书的其他比较难理解的知识点也可以采用这种学习方法。这种方法，称为斯科普（Skip）学习法。

第 5 章　制作流水灯

本章 Python 极客将会利用一些零件制作动态的流水灯，本章案例同样很简单，就是通过 Python 代码控制 PyBox 上的 LED 灯依次滚动闪烁。

做完这个实验读者会发现，只需要简单的几行代码就能玩转 PyBox 上的 LED 灯了！

黑夜里闪烁着五颜六色的流水灯还是很酷的。

零件清单

本章需要的零件如下：

（1）PyBox 极客迷你小电脑一台.

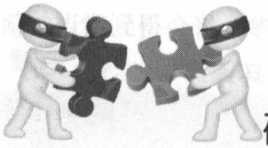
（2）Micro USB 电脑连接线一根。

编写程序代码

本案例的相关脚本文件名为：TPY_002.py。核心代码如下。

```
leds = [pyb.LED(i) for i in range(1,5)]
for l in leds:
    l.off()
```

```
n = 0
try:
   while True:
      n = (n + 1) % 4
      leds[n].toggle()
      pyb.delay(50)
finally:
    for l in leds:
       l.off()
```



硬件连接

本案例没有额外零件，直接用 Micro USB 连接线，把 PyBpx 迷你电脑和 PC 的 USB 接口连接即可。

有图为证

用 Micro USB 把 PyBpx 迷你小电脑和 PC 的 USB 接口连接，如图 5-1 所示。

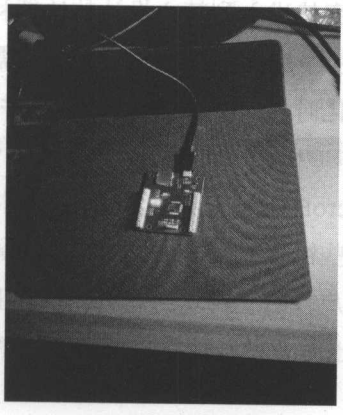


图 5-1　硬件连接图

注意事项

目前我们使用了板上的单只 LED 灯，而实际上总共有 4 只灯可供使用，可以为每一只 LED 灯创建一个对象，并分别控制它们，我们将声明一个便于理解的列表（List）形式：

```
leds = [pyb.LED(i) for i in range(1,5)]
```

如果没有使用 1、2、3、4 的数字作为 pyb.LED( )的形参，将会得到错误的信息。接下来我们将添加每一个 LED 灯点亮、熄灭的无限循环：

```
n = 0
while True:
  n = (n + 1) % 4
  leds[n].toggle()
  pyb.delay(50)
```

在这里，n 代表了当前的 LED 灯，每次循环执行后我们可以得到下一个 n 的值（求余符号%保证了 n 的值在 0 和 3 之间）。

然后，就可以控制第 n 颗 LED 灯的翻转点亮、熄灭了。

执行该程序，将可见一排的 LED 灯同时点亮和熄灭。

你可能会发现一旦停下当前脚本的运行并重新启动，开发板上的 LED 灯将从之前运行的单灯切换状态，进入多灯切换状态，而且每个灯泡颜色不同，五颜六色，轮流闪烁，给人一种突然进入到精心设计的迪斯科歌舞厅的感觉。

由于本案例 main.py 程序是无限循环模式，所以需要人工手动关闭这些 LED 灯，方法是：在脚本初始化时关闭所有的 LED 灯，并使用 try/finally 块的方式解决这个问题。

当键入 CTRL-C，MicroPython 将产生一个 VCP 中断异常。

异常，通常意味着某些错误的情况，可以通过 try:command 指令“抓取”一个异常。这种情况属于用户打断了脚本的运行，所以我们不需要抓取错误而是简单地告诉 MicroPython 退出时要做些什么。最终的程序块确保了所有的 LED 灯熄灭，如下所示。

```
try:
    while True:
        n = (n + 1) % 4
        leds[n].toggle()
        pyb.delay(50)
finally:
    for l in leds:
        l.off()
```

注意以上代码的 try...finally 容错语句，有关的细节，请大家自己查询 Python 语法资料。

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，将 PyBox 盘符 TPFLASH 上面的 main.py 文件用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄灭，按下“Reset”（RST）复位按钮，运行 main.py 文件中的程序代码。

成功时刻

运行 TPY_002.py 程序成功后，电流会在 PyBox 板上跳“迪斯科（disco）”，即四个 LED 灯来回闪烁。

进阶学习

特殊的第四颗灯。

蓝色的 LED 灯比较特别，可以在让其点亮、熄灭的同时，通过 iniensity( )的方法控制其亮度，其亮度值在 0～255 之间。以下的脚本实现了蓝色的 LED 灯循环渐亮，然后熄灭的功能。

```
led = pyb.LED(4)
intensity = 0
while True:
    intensity = (intensity + 1) % 255
    led.intensity(intensity)
    pyb.delay(20)
```

也可以对其他 LED 灯调用 instensity( )的方法，不过其他 LED 灯只能被熄灭或被点亮，0 值将使之熄灭，其他值只能点亮该 LED 灯。

注意要点

- 点亮 LED 1 灯闪红色光 Red。
- 点亮 LED 2 灯闪绿色光 Green。
- 点亮 LED 3 灯闪橘黄色光 Orange。
- 点亮 LED 4 灯闪蓝色光 Blue。

第 6 章　点亮心形 8×8 点阵

当你在地铁站、公交站以及一些公共场所看到一些指示标志时，有没有想过自己用点阵做一个很酷的小项目呢？

比如做一个心形的点阵，在情人节那天向你心目中的“女神”表白。

本章介绍，如何使用 Python 程序，点亮 8×8LED 点阵，形成一个漂亮的心形。

零件清单

本章需要的零件很简单，如图 6-1 所示，元件清单如下：

（1）8×8 LED 点阵一个。

（2）PyBox 板子一块。

（3）数据线一条。

（4）两头都是母线的杜邦线两条。

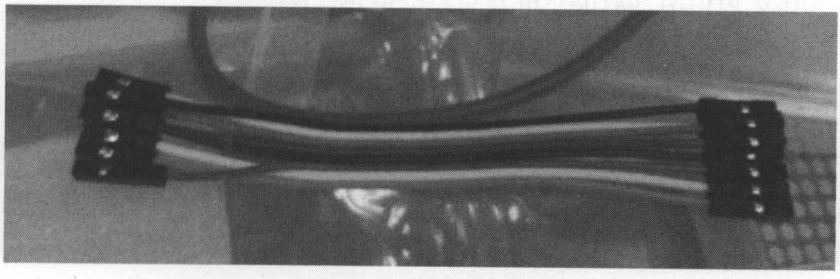
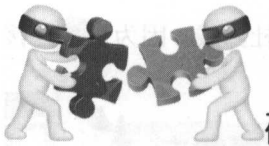


图 6-1　元件清单

编写程序代码

按照上面的步骤做完以后通电、编写 main.py 文件，即可显示你想要的字符或者图案，本案例的相关脚本文件名为：TPY_003.py。以下代码是在 8×8 点阵上显示心形图案，具体代码如下：

```
import pyb
from pyb import Pin
x_PIN = [Pin(i, Pin.OUT_PP) for i in ['X1','X2','X3','X4',
'X5','X6','X7','X8']]
y_PIN = [Pin(i, Pin.OUT_PP) for i in ['Y1','Y2','Y3','Y4',
'Y5','Y6','Y7','Y8']]
hanzi=['11111111','11011101','10001000','10000000','10000000','1
1000001','11100011','11110111']
def displayLED():
    flag=0
    for x_ in range(0,8):
        for b in range(0,8):
            print(b)
            if b!=flag:
                x_PIN[b].value(0)
        li_l = hanzi[x_]
        y_PIN[0].value(int(li_l[:1]))
        y_PIN[1].value(int(li_l[1:2]))
        y_PIN[2].value(int(li_l[2:3]))
        y_PIN[3].value(int(li_l[3:4]))
        y_PIN[4].value(int(li_l[4:5]))
        y_PIN[5].value(int(li_l[5:6]))
        y_PIN[6].value(int(li_l[6:7]))
        y_PIN[7].value(int(li_l[7:8]))
        x_PIN[flag].value(1)
        flag=flag+1
        pyb.delay(2)
while 1:
    displayLED()
```



硬件连接

如图 6-2 所示，LED 点阵后面有两排针脚，一排以 1 开头，即 1～8 针脚，一排以 9 开头，即 9～16 针脚，图中⑨、⑭、⑧、⑫、①、⑦、②、⑤为针脚对应的数字。当 ROW 的⑨、⑭、⑧、⑫、①、⑦、②、⑤为高电平，COL PIN NO. 为低电平的时候，LED 灯即全部被点亮，为了方便操作行和列，可以将 ROW 的 8 个引脚接到 PyBox 的 *X*1～*X*8，COL 的 8 个引脚接到 PyBox 的 *Y*1～*Y*8，这样就可以通过控制 *X* 引脚、*Y* 引脚来决定点亮哪个灯、熄灭哪个灯，并且想显示任何字符都没有问题，快来试试吧。

如图 6-2 所示，*X*1～*X*8 引脚接点阵屏的数字 1～8 针脚，*Y*1～*Y*8 引脚接点阵屏的数字 9～16 针脚。点阵屏上面两排针脚，一排以 1 标注，表示依次是 1～8，同理另外一排是 9～16。

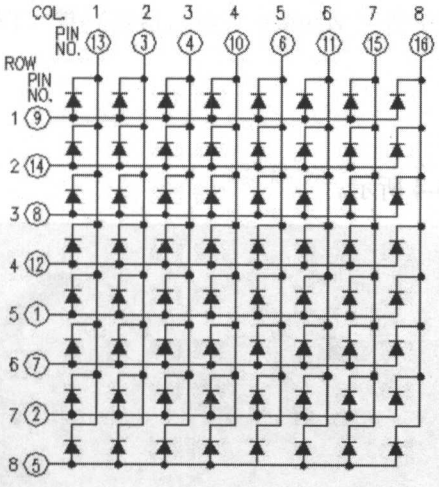


图 6-2　8×8 LED 点阵屏

我们先来将 *X*1 接入 ROW 的 1，1 旁边的数字是⑨，说明 *X*1 和实物的第 9 个引脚连接。圆圈内的数字表示的是排针的编号，也就是说点阵屏的 1 其实是第 9 号排针，依此类推。

连接 TYPBoard 和 8×8 点阵屏，需要两头都是母线的杜邦线。因为需要连接 16 个引脚，所以需要 16 根杜邦线，如图 6-3 所示。

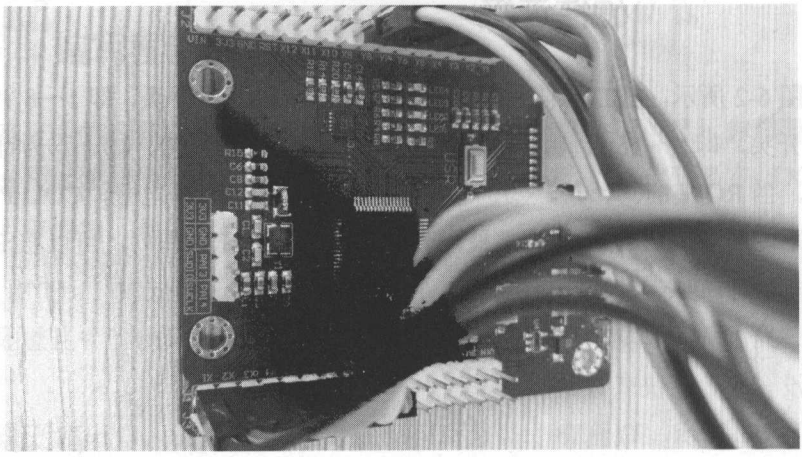


图 6-3　连接 16 个引脚需要 16 根杜邦线

有图为证

硬件连接图如图 6-4 所示。

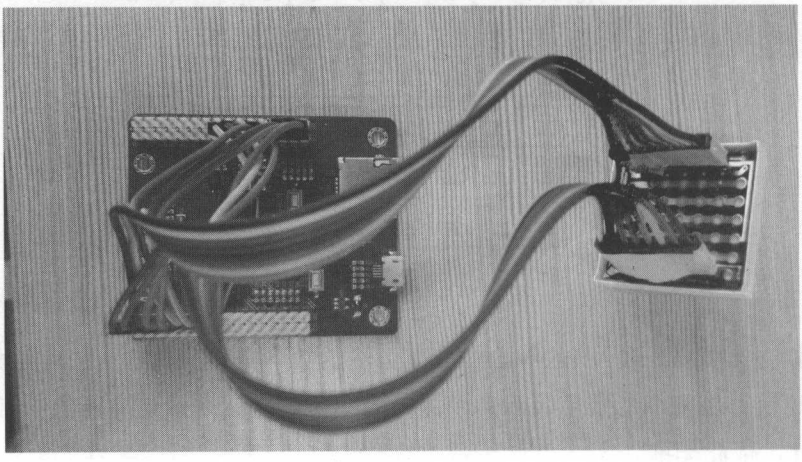


图 6-4　硬件连接图

注意事项

注意杜邦线与 PyBox 的插入与拔出。

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄灭，按下“Reset”（RST）复位按钮，运行 main.py 文件中的程序代码。

成功时刻

点亮心形 8×8 点阵如图 6-5 所示。

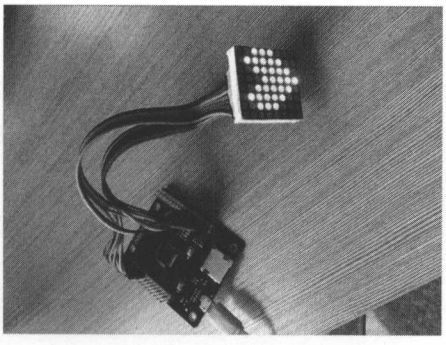


图 6-5　点亮心形 8×8 点阵

注意要点

实验目的：

（1）学习在 PC 机系统中扩展简单 I/O 接口的方法。

（2）进一步学习编制数据输出程序的设计方法。

（3）学习 8×8 点阵与 PyBox 的接线方法，点亮点阵。

第 7 章　模拟红绿灯教程

PyBox 搭配传感器和 LED 灯后，具有非常广泛的应用功能，比如控制红绿灯等。本章将使用 Python 程序依次点亮红、黄、绿三个 LED 灯，模拟现实交通中的红绿灯。

零件清单

本章需要的零件如下：

（1）220 欧电阻一个。

（2）红色 LED 数码管一个。

（3）面包板一块。

（4）PyBox 板子一块。

（5）数据线一条。

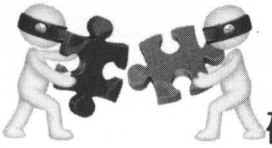
（6）红、黄、绿三个 LED 灯。

（7）两头都是母线的杜邦线两条。

编写程序代码

本案例的程序文件名是：TPY_004a.py，通过以下代码点亮 LED 灯，具体代码如下。

```
import pyb
led1 = pyb.Pin("Y1",pyb.Pin.OUT_PP)
led2 = pyb.Pin("Y2",pyb.Pin.OUT_PP)
led3 = pyb.Pin("Y3",pyb.Pin.OUT_PP)
While True:
    led1.value(1)
    led2.value(1)
    led3.value(1)
```



硬件连接

将三个 LED 灯插在面包板上，LED 负极插入面包板的负极（横向插孔），正极插入面包板的纵向插，将 220 欧电阻插入面包板的负极（横向插孔）和纵向插孔中，将 LED 灯的正极分别与 PyBox 的引脚连接起来，只需要三个引脚即可，笔者用的是 *Y*1、*Y*2、*Y*3 三个引脚，将三个 LED 灯的正极通过杜邦线连接到 PyBox 的 *Y*1，、*Y*2、*Y*3 的引脚上，然后将电阻纵向插孔用杜邦线接到 PyBox 的 GND 引脚，在 main.py 文件中将 *Y*1、*Y*2、*Y*3 引脚的电平拉高，即可看到三个灯同时亮起来。

有图为证

红、黄、绿三个灯同时亮起来如图 7-1 所示。

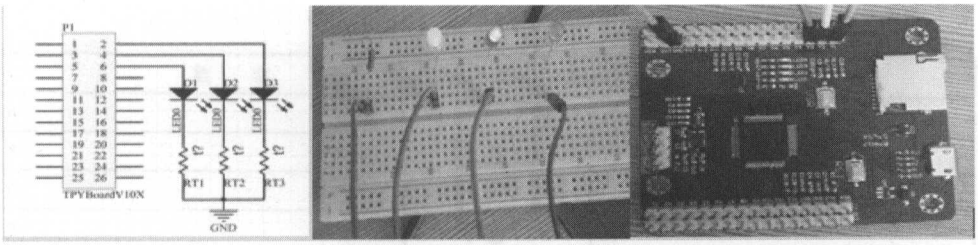


图 7-1　红、黄、绿三个灯同时亮起来

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件，用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄灭，按下“Reset”（RST）复位按钮，运行 main.py 文件中的程序代码。

成功时刻

成功运行程序后，可以看到红、黄、绿三个灯同时亮起来。

进阶学习

点亮 SM42056 数码管如图 7-2 所示。

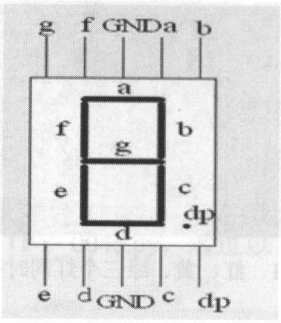


图 7-2　SM42056 数码管

SM42056 是 0.56 英寸一位共阴/红色 LED 数码管。一共 10 个引脚。当小数点在你的右下角时，上面一排 5 个引脚，从左至右依次为 g、f、GND（地线）、a、b，下面一排 5 个引脚，从左至右依次为 e、d、GND（地线）、c、dp、如图 7-2 所示。

要想让数码管亮起来，只需要将 g、f、a、b、e、d、c、dp（在这用不到）在 main.py 中拉高电平，把数码管的 GND（地线），与 PyBox 的 GND 引脚接起来，这样就会显示为数字 8，如图 7-3 所示。

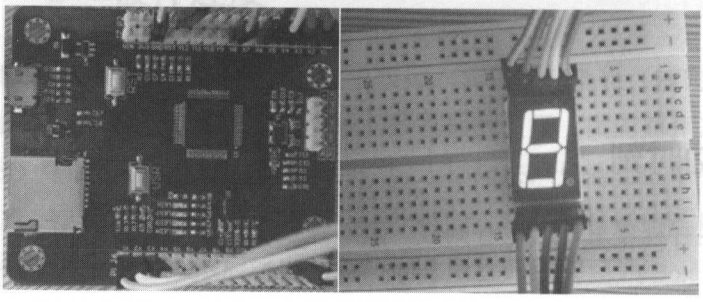


图 7-3　硬件连接图

如表 7-1 所示为 0-9 数字对应针脚的高电平，即对应 PyBox 的引脚拉高电平。

表 7-1　0-9 数字对应针脚的高电平

数　字	高电平针脚
0	a、b、c、d、e、f
1	e、f
2	a、b、g、e、d
3	a、b、g、c、d

续表

数　字	高电平针脚
4	b、c、g、f
5	a、f、g、c、d
6	a、f、e、d、c、g
7	a、b、c
8	a、b、c、d、e、f、g
9	a、b、c、d、f、g

注意要点

按照以上步骤安装完以后就可以通电了，本案例的程序文件名是：TPY_004b.py，修改 main.py 文件，即可让 LED 灯随着数码管的变化而变化，具体代码如下。

```
import pyb

led1 = pyb.Pin("Y1",pyb.Pin.OUT_PP)
led2 = pyb.Pin("Y2",pyb.Pin.OUT_PP)
led3 = pyb.Pin("Y3",pyb.Pin.OUT_PP)
x1 = pyb.Pin("X1",pyb.Pin.OUT_PP)
x2 = pyb.Pin("X2",pyb.Pin.OUT_PP)
x3 = pyb.Pin("X3",pyb.Pin.OUT_PP)
x4 = pyb.Pin("X4",pyb.Pin.OUT_PP)
x5 = pyb.Pin("X5",pyb.Pin.OUT_PP)
x6 = pyb.Pin("X6",pyb.Pin.OUT_PP)
x8 = pyb.Pin("X8",pyb.Pin.OUT_PP)

def six():
    x1.value(1)
    x2.value(1)
    x3.value(1)
    x5.value(1)
    x6.value(1)
    x8.value(1)
    pyb.delay(1000)
```

```
    x1.value(0)
    x2.value(0)
    x3.value(0)
    x6.value(0)
    x5.value(0)
    x8.value(0)

def nine():
    x1.value(1)
    x2.value(1)
    x3.value(1)
    x4.value(1)
    x5.value(1)
    x8.value(1)
    pyb.delay(1000)
    x1.value(0)
    x2.value(0)
    x3.value(0)
    x4.value(0)
    x5.value(0)
    x8.value(0)

def eight():
    x1.value(1)
    x2.value(1)
    x3.value(1)
    x4.value(1)
    x5.value(1)
    x6.value(1)
    x8.value(1)
    pyb.delay(1000)
    x1.value(0)
    x2.value(0)
    x3.value(0)
    x4.value(0)
    x5.value(0)
    x6.value(0)
    x8.value(0)

def zero():
    x2.value(1)
    x3.value(1)
```

```
    x4.value(1)
    x5.value(1)
    x6.value(1)
    x8.value(1)
    pyb.delay(1000)
    x2.value(0)
    x3.value(0)
    x4.value(0)
    x5.value(0)
    x6.value(0)
    x8.value(0)

def seven():
    x3.value(1)
    x4.value(1)
    x8.value(1)
    pyb.delay(1000)
    x3.value(0)
    x4.value(0)
    x8.value(0)

def five():
    x1.value(1)
    x2.value(1)
    x3.value(1)
    x5.value(1)
    x8.value(1)
    pyb.delay(1000)
    x1.value(0)
    x2.value(0)
    x3.value(0)
    x5.value(0)
    x8.value(0)

def four():
    x1.value(1)
    x2.value(1)
    x4.value(1)
    x8.value(1)
    pyb.delay(1000)
    x1.value(0)
    x2.value(0)
```

```
    x4.value(0)
    x8.value(0)

def three():
    x1.value(1)
    x3.value(1)
    x4.value(1)
    x5.value(1)
    x8.value(1)
    pyb.delay(1000)
    x1.value(0)
    x4.value(0)
    x3.value(0)
    x5.value(0)
    x8.value(0)

def two():
    x1.value(1)
    x3.value(1)
    x4.value(1)
    x5.value(1)
    x6.value(1)
    pyb.delay(1000)
    x1.value(0)
    x3.value(0)
    x4.value(0)
    x5.value(0)
    x6.value(0)

def one():
    x2.value(1)
    x6.value(1)
    pyb.delay(1000)
    x2.value(0)
    x6.value(0)

while True:
    led1.value(1)
    nine()
    eight()
    seven()
    six()
```

```
    five()
    four()
    three()
    two()
    one()
    zero()
    led1.value(0)
    led2.value(1)
    nine()
    eight()
    seven()
    six()
    five()
    four()
    three()
    two()
    one()
    zero()
    led2.value(0)
    led3.value(1)
    three()
    two()
    one()
    zero()
    led3.value(0)
```

实际效果演示地址：

http://v.youku.com/v_show/id_XMTY1MzY5NDExNg==.html。

第 8 章　DIY 数字温度计

目前，在很多工业控制领域，温度监控普遍利用热敏电阻组成的测温电路来实现测温。

DS18B20 数字温度传感器，是 Dallas 公司生产的 1-Wire，即单总线器件，具有线路简单、体积小的特点，如图 8-1 所示。

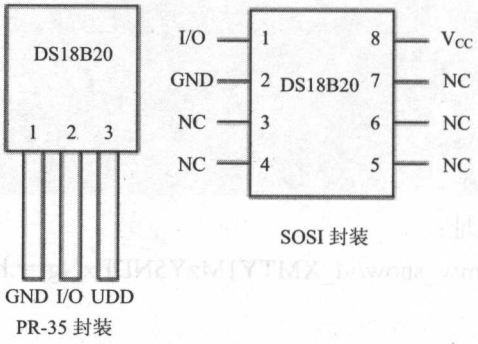


图 8-1　DS18B20 数字温度传感器

因此用它组成的测温系统其线路将非常简单，一根通信线就可以悬挂很多这样的数字温度传感器，十分方便。

本章，我们将使用 Python 程序操纵 PyBox 迷你小电脑和 DS18B20 数字温度传感器，联合监测当前环境下的温度。

零件清单

本章需要的零件如下：

（1）PyBox 板子一块。

（2）数据线一条。

（3）杜邦线若干条。

（4）8×8 LED 点阵一个。

（5）DS18B20 温度传感器一个。

编写程序代码

本章主要学习 DS18B20 的接线方法，并检测当前温度。本案例的相关脚本文件名为：TPY_005.py，具体的源代码如下。

```
#main.py
import pyb
from pyb import Pin
from ds18x20 import DS18X20
Pin("Y11",Pin.OUT_PP).low()#GND
Pin("Y9",Pin.OUT_PP).high()#VCC
pyb.delay(100)
DQ=DS18X20(Pin('Y10'))#DQ
while True:
      tem = DQ.read_temp()
      print(tem)
      pyb.delay(1000)
```

有图为证

如图 8-2 所示为硬件连接图。

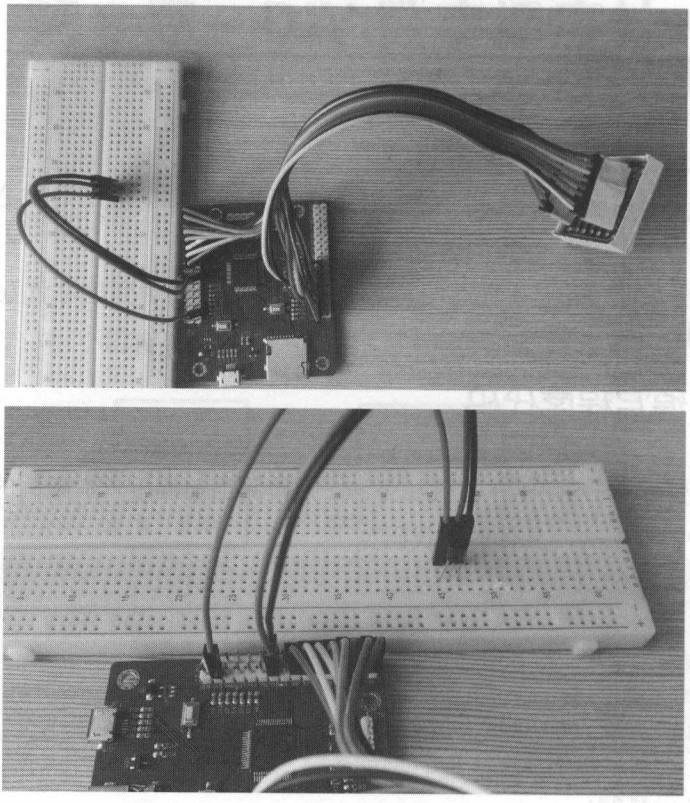


图 8-2　硬件连接图

注意事项

先看一下 DS18B20 针脚含义，如图 8-3 所示。PyBox 的针脚与 DS18B20 的

针脚对应的关系，如图 8-4 所示。针脚编号如图 8-5 所示。

DS18B20 的接线完成后，在 MicroPython 的源码目录中，进入 drivers\onewire\ 目录，然后将目录下的文件 ds18x20.py 和 onewire.py 复制到 PYBFLASH 磁盘的根目录。复制文件后要安全退出磁盘，然后重新接入（不然找不到该文件），即可运行 main.py 文件了，print 输出 DS18B20 温度传感器读取的温度数据，即可用 Putty 看到当前的温度。

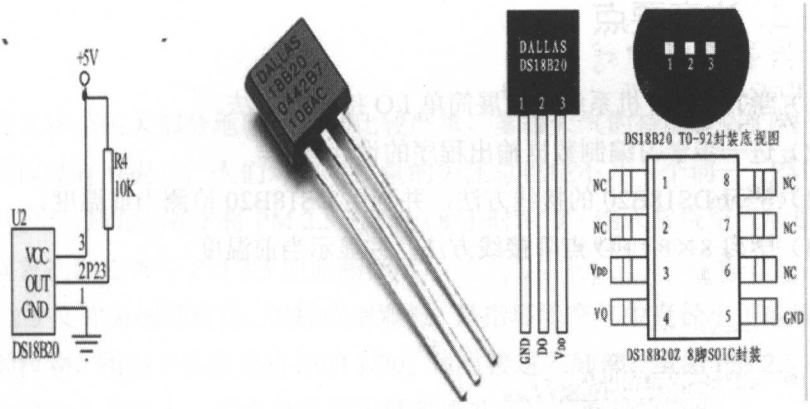


图 8-3　DS18B20 针脚含义

```
TPYBoard     DS18B20
---------------
# 3V3  or any Pin  => VDD
# any  Pin        =>  DO
#  GND            => GND
```

图 8-4　PyBox 的针脚与 DS18B20 的针脚对应的关系

```
TPYBoard     LCD5110
-------------------------
3.3v      =>  VDD
GND       =>  GND
Y10       =>  DO
```

图 8-5　针脚编号

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件，用程序代码编辑软件打开，把新的程序代码复制到 main.py

文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄灭，按下“Reset”（RST）复位按钮，运行的 main.py 文件中的程序代码。

注意要点

（1）学习在 PC 机系统中扩展简单 I/O 接口的方法。

（2）进一步学习编制数据输出程序的设计方法。

（3）学习 DS18B20 的接线方法，并利用 DS18B20 检测当前温度。

（4）学习 8×8 LED 点阵接线方法，并显示当前温度。

第 9 章　PM 2.5 检测仪

最近几年全国大部分地区雾霾都比较严重，雾霾天气的持续让人们对空气质量的关注程度有所提升，人们对空气质量的关注总也绕不开一个词——PM 2.5。

《环境空气质量标准》将 PM 2.5、臭氧（8 小时浓度）纳入常规空气质量评价，这是我国首次制定关于 PM 2.5 的监测标准。

PM 2.5 又称为细颗粒物、细粒或细颗粒，是指环境空气中直径小于或等于 2.5 微米的颗粒物，相当于头发丝直径的 1/20，可直接进入肺部。虽然 PM 2.5 在地球大气成分中的含量很少，但它对空气质量和能见度等有重要的影响。

本章，我们使用 PyBox 迷你小电脑和 PM 2.5 粉尘传感器，来检测当前环境下的 PM 2.5 值。

自己动手制作 PM 2.5 检测仪，当空气质量较差或者严重污染的时候，提醒家人、同学或者同事，尽量减少户外活动，以减少吸入细颗粒物。

零件清单

制作 PM 2.5 检测仪需要的零件如下：

（1）PM 2.5 粉尘传感器 1 个，用来检测 PM 2.5（细颗粒物），TXD 串口输出。

（2）PyBox 开发板 1 块，主要用作主控开发板，读入传感器数据。

（3）LCD5110 显示屏 1 个，主要用来显示检测到的信息。

（4）杜邦线若干条。

（5）数据线一条。

编写程序代码

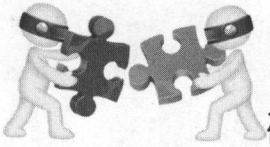
在 main.py 文件中，首先定义串口，其次打开串口接收采样数据，最后关闭串口，并且处理采样数据及显示，依次循环。本案例的相关脚本文件名为：TPY_006.py，具体的程序代码如下。

```
#main.py
import pyb
import upcd8544
from machine import SPI,Pin
from pyb import UART
from ubinascii import hexlify
from ubinascii import *

leds = [pyb.LED(i) for i in range(1,5)]
P,L,SHUCHU=0,0,0
#A 比例系数，在北方一般使用 800～1000，南方空气好一些，一般使用 600～800，这还与你使用的传感器灵敏度有关，需要自己测试完再定下来。
A=800
#G 为固定系数，是为了把串口收到的数据转换成 PM 标准值
G=1024/5
SPI = pyb.SPI(1) #DIN=>X8-MOSI/CLK=>X6-SCK
#DIN =>SPI(1).MOSI 'X8' data flow (Master out, Slave in)
#CLK =>SPI(1).SCK  'X6' SPI clock
RST   = pyb.Pin('Y10')
CE    = pyb.Pin('Y11')
DC    = pyb.Pin('Y9')
LIGHT  = pyb.Pin('Y12')
lcd_5110 = upcd8544.PCD8544(SPI, RST, CE, DC, LIGHT)
u2 = UART(2, 2400)
count_=0
def ChangeLEDState(num_):
    global leds
    len_=len(leds)
```

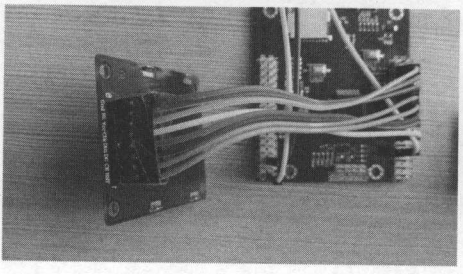
```
    for i in range(0,len_):
        if i!=num_:
            leds[i].off()
        else:
            leds[i].on()
while True:
    u2.init(2400, bits=8, parity=None, stop=1)
    pyb.delay(80)
    Quality='DATA NULL'
    if(u2.any()>0):
        u2.deinit()
        _dataRead=u2.readall()
        #R 代表截取数据的起始位
        R=_dataRead.find(b'\xaa')
        #R>-1 代表存在起始位，长度大于起始位位置+2
        if R>-1 and len(_dataRead)>(R+2):
            P=_dataRead[R+1]
            L=_dataRead[R+2]
            #把串口收到的十六进制数据转换成十进制
            SHI=P*256+L
            SHUCHU=SHI/G*A
        if(SHUCHU<35):
            Quality = 'Excellente'
            print('环境质量:优','PM2.5=',SHUCHU)
            count_=1
        elif(35<SHUCHU<75):
            Quality = 'Good'
            print('环境质量: 良好','PM2.5=',SHUCHU)
            count_=1
        elif(75<SHUCHU<115):
            Quality = 'Slightly-polluted'
            print('环境质量: 轻度污染 ','PM2.5=',SHUCHU)
            count_=3
        elif(115<SHUCHU<150):
            Quality = 'Medium pollution'
            print('环境质量: 中度污染 ','PM2.5=',SHUCHU)
            count_=2
        elif(150<SHUCHU<250):
            Quality = 'Heavy pollution'
            print('环境质量: 重度污染 ','PM2.5=',SHUCHU)
```

```
            count_=0
        elif(250<SHUCHU):
            Quality = 'Serious pollution'
            print('环境质量：严重污染 ','PM2.5=',SHUCHU)
            count_=0
    ChangeLEDState(count_)
    lcd_5110.lcd_write_string('AQI Level',0,0)
    lcd_5110.lcd_write_string(str(Quality),0,1)
    lcd_5110.lcd_write_string('PM2.5:',0,2)
    lcd_5110.lcd_write_string(str(SHUCHU),0,3)
```



硬件连接

如图 9-1 所示为硬件连接图。



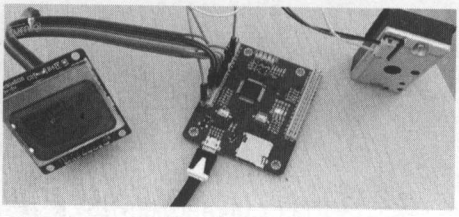


图 9-1　硬件连接图

注意事项

传感器上一共 6 根线，依次是 GND、VCC、NC、NC、RX、TX，其中只用 3 根线。电源（GND、VCC）和串口（TX）、传感器与 TPYBorad 的接线参照图 9-1 所示，笔者使用的串口为 UART(2) is on: (TX, RX) = (X3, X4) = (PA2, PA3)，因为只需要将数据传到 PTYBoard，所以只用到 RED，即 PTYBoard 的 X4 引脚。如图 9-2 所示。

图 9-2　数据传到 PTYBoard

先看一下 LCD5110 针脚的含义吧（注意：LCD5110 的针脚有些特别），PyBox 的针脚与 LCD 5110 的针脚对应关系如下：

```
TPYBoard    LCD5110   memo
----------------------------
# any  Pin    => RST     Reset pin (0=reset, 1=normal)
# any  Pin    => CE      Chip Enable (0=listen for   input,  1=ignore input)
# any  Pin    => DC      Data/Command (0=commands, 1=data)
# MOSI     => DIN     data flow (Master out, Slave in)
# SCK      => CLK     SPI clock
# 3V3 or any Pin   => VCC     3.3V logic voltage (0=off, 1=on)
# any Pin     => LIGHT   Light (0=on, 1=off)
# GND      => GND
```

针脚编号如下：

```
TPYBoard    LCD5110   memo
----------------------------
Y10      => RST     Reset pin (0=reset, 1=normal)
Y11      => CE      Chip Enable (0=listen for input, 1=ignore input)
Y9       => DC      Data/Command (0=commands, 1=data)
X8       => DIN     data flow (Master out, Slave in)
X6       => CLK     SPI clock
VCC
Y12      => LIGHT   Light (0=on, 1=off)
GND
```

连接线如图 9-3 所示。

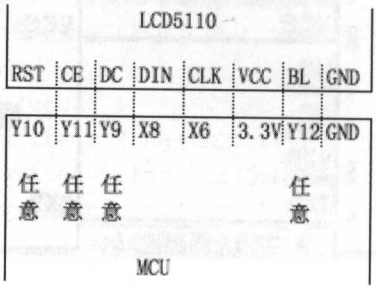


图 9-3　连接线

将 PM 2.5 粉尘传感器以及 LCD5110 显示屏与 PTYBoard 连接起来，硬件连接完毕，如图 9-4 所示。

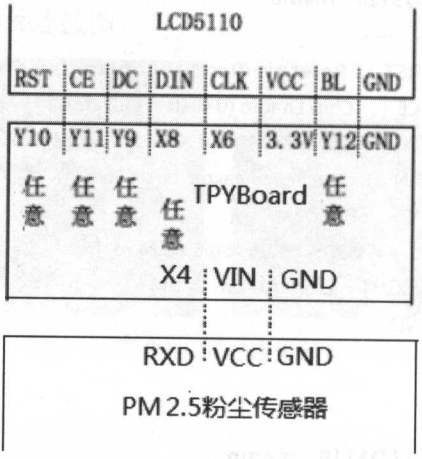


图 9-4　硬件连接完成图

开始运行

连接线接好以后，导入 font.py 文件和 upcd8544.py 文件（主要用于 LCD5110 显示数据），再运行 main.py 文件，即可看到当前的空气质量等级以及 PM 2.5 的

浓度值了。

成功时刻

做好的 PM 2.5 粉尘传感器如图 9-5 所示。

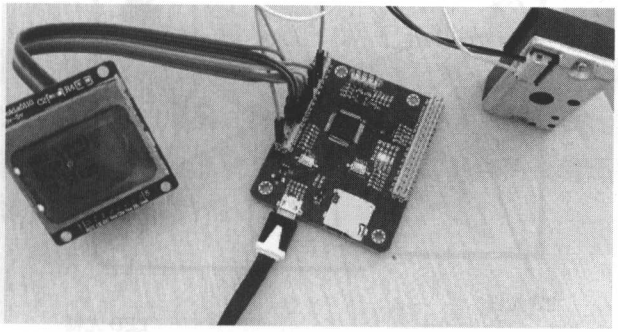
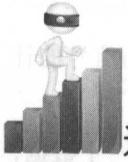


图 9-5　做好的 PM 2.5 粉尘传感器



进阶学习

PM 2.5 粉尘传感器工作原理：PM 2.5 粉尘传感器是根据光的散射原理来开发的，微粒和分子在光的照射下会产生光的散射现象，与此同时，还吸收部分照射光的能量。

当一束平行单色光入射到被测颗粒场时，会受到颗粒周围散射和吸收的影响，光强将被衰减。如此一来便可求得入射光通过待测浓度场的相对衰减率。而相对衰减率的大小基本上能线性反应待测场灰尘的相对浓度。光强的大小和经光电转换的电信号强弱成正比，通过测得电信号就可以求得相对衰减率，进而就可以测定待测场里灰尘的浓度。

在传感器的中间有一个洞，这个洞可以让空气在里面流通。在洞的两个边缘，

一面安装一个激光发射器，另一面安装激光接收器。这样一来，空气流过这个小洞，空气里的颗粒物就会挡住激光，从而产生散射，另一面的接收器，依据接收到的激光强度来发出不同的信号（其实就是输出不同的电压值）。这样一来，空气里的颗粒物越多，输出的电压越高，颗粒物越少，输出的电压越低。

内部结构仿真图如图 9-6 所示。

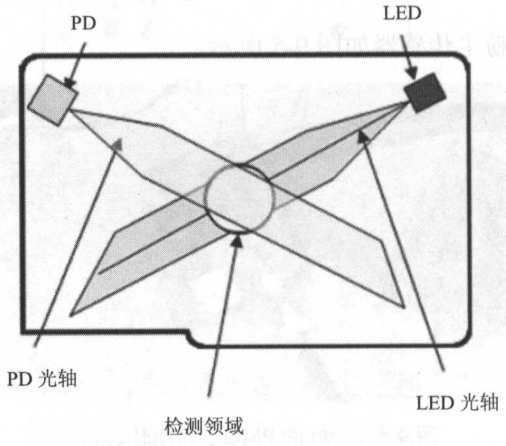


图 9-6　内部结构仿真图

上面说了传感器的原理，下面说说它传出来的信号和对于接收到的信号的计算。

传感器的输出数据是靠串口进行传输的，通过串口每 3～4ms 发送一个数据，数据类型大致是“0X00”的 16 进制。每次数据会以“0XAA”作为起始端，以“0XFF”作为结束端。共 7 个数据位：起始位、结束位、数据高位、数据低位、数据高校验位、数据低校验位和校验位，数据格式大致如表 9-1 所示。

表 9-1　数据发送格式

起始位	Vout(H)	Vout(L)	Vref(H)	Vref(L)	校验位	结束位
0xaa	如：0x01	如：0x3a	如：0x00	如：0x7a	如：0xd0	0xff

其中校验位长度=Vout（H）+Vout（L）+Vref（H）+Vref（L）

数据的组成一共有 7 个数据位，但是只有 Vout（H）和 Vout（L）这两个数据才是我们真正所需要的，依照这两个数据计算出来串口输出的数字数据，通过数模转换公式来计算出输出的电压。进一步通过比例系数计算出空气中颗粒物的数量。下面说一下怎么计算。

传感器输出的数据分为高位和低位，其中 Vout（H）为高位，Vout（L）为低位。因为串口传进来的 Vout（H）和 Vout（L）是 16 进制的，第一步先转化成 10 进制。然后根据这两个输出值的 10 进制数计算出串口输出数值的电压。

公式如下（其中 Vout（H）和 Vout（L）已转化为 10 进制）：

```
Vout=（Vout（H）*256+Vout（L））/1024*5
```

这样就算出来了输出的电压，再根据比例系数 A，就可以计算出空气中的颗粒物的值。比例系数 A 的值一般是在 800～1000，具体的数值还要根据你的传感器的精度、准确度和误差值进行确定。

注意要点

PM 2.5 粉尘传感器的采样频率是非常高的，一般 3～4ms 发送一个 16 进制的采样数据，也就是说传感器通电（接通 VCC 和 GND）后，每隔 3～4ms 发送一个 16 进制的采样数据，作为一个检测仪来说这么高的采样频率显然没有必要。

PyBox 通过串口接收粉尘传感器数据，使用串口当然要先定义串口，打开串口就可以接收串口数据，关闭串口就停止接收数据，自由控制 PM 2.5 粉尘传感器的采样频率。

第 10 章　智能扫雷仪

扫雷游戏大概是国民普及度最高的电脑游戏了，许多老年人用电脑打字都很困难，但是玩起扫雷时鼠标飞舞。本章将使用 PyBox 迷你小电脑和 LJ12A3-4-Z/BX 金属接近开关，来检测开关附近是否有金属，以模拟实际环境中的智能扫雷。

零件清单

本章需要的零件如下：

（1）PyBox 板子一块。

（2）LJ12A3-4-Z/BX 金属接近开关一个。

（3）面包板一块。

（4）发光二极管一个。

（5）数据线一条。

（6）杜邦线若干条。

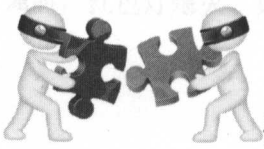
编写程序代码

本案例的相关脚本文件名为：TPY_007.py，具体代码如下。

```
import pyb
from machine import Pin

y1 = Pin('Y1', Pin.IN)
x1 = Pin('X1', Pin.OUT_PP)

while 1:
    #无金属时
    if y1.value() == 1 :
        print(y1.value())
        x1.value(0)
    #有金属时
    else:
        print(y1.value())
        x1.value(1)
```



硬件连接

将接近开关线接好后，PyBox 开发板通过 *Y1* 针脚收集金属开关传递过来的数字信号，通过这个信号，让开发板控制自动门开关、报警等，这只是做了一个简单易懂的应用，点亮红色 LED 发光二极管。

有图为证

硬件连接图如图 10-1 所示。

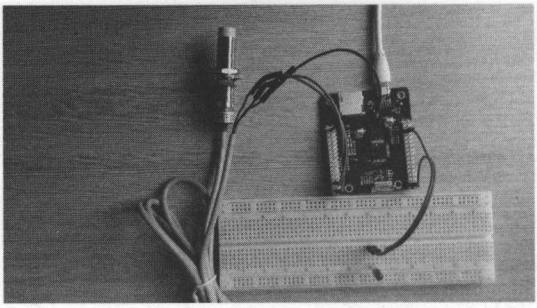


图 10-1　硬件连接图

注意事项

LJ12A3-4-Z/BX 接近开关工作原理如图 10-2 所示。

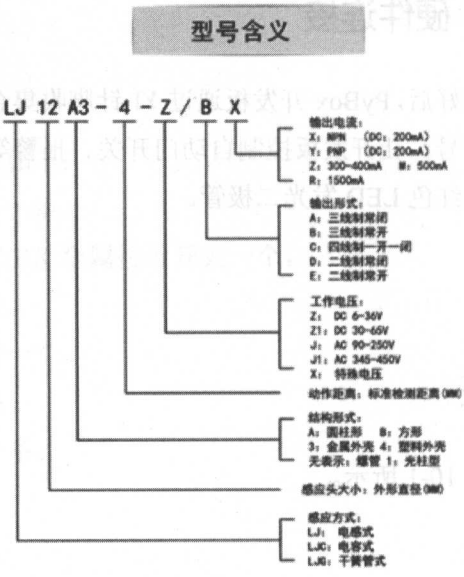


图 10-2　LJ12A3-4-Z/BX 接近开关工作原理

接近开关既有行程开关、微动开关的特性，也具有传感性，同时具有动作可

靠、性能稳定、频率响应快、应用寿命长、抗干扰能力强、防水防震、耐腐蚀等特点。

接近开关的输出信号是输出数字信号，即当没有金属靠近时，输出 1，当有金属时，输出 0。通过探头检测是否有金属，然后将检测的数字传递到 PyBox，PyBox 做出相应的判断，广泛应用于机床、冶金、化工、航天航空、轻纺与印刷等行业；在日常生活中，可用于宾馆、饭店、车库的自动门、自动热风机等；在安全防盗方面，如资料档案、财会、金融、博物馆、金库等重地，通常都装有各种接近开关组成的防盗装置。

我们只需要正极（灰线）连接 PyBox 的 VIN 引脚，负极（蓝线）接 PyBox 的 GND 引脚，黑线（输出信号）连接 PyBox 的 IO 针脚，在这用到 *Y*1 针脚，连接完毕后，当有金属靠近时，接近开关本身自带的红色灯就会亮起来，当远离金属时，红色灯熄灭。如图 10-3 所示。

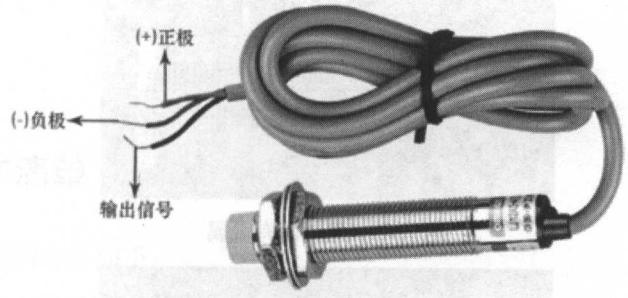


图 10-3　连接 PyBox 的 VIN 引脚

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件，用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄

灭，按下“Reset”（RST）复位按钮，运行 main.py 文件当中的程序代码。

成功时刻

开关周围没有 LJ12A3-4-Z/BX 金属时，如图 10-4 所示，LJ12A3-4-Z/BX 金属接近开关时，如图 10-5 所示。

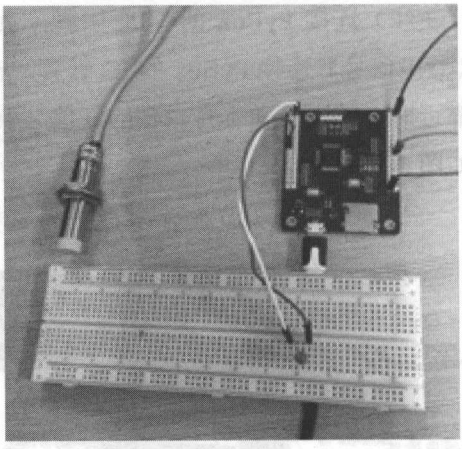


图 10-4　周围没有金属

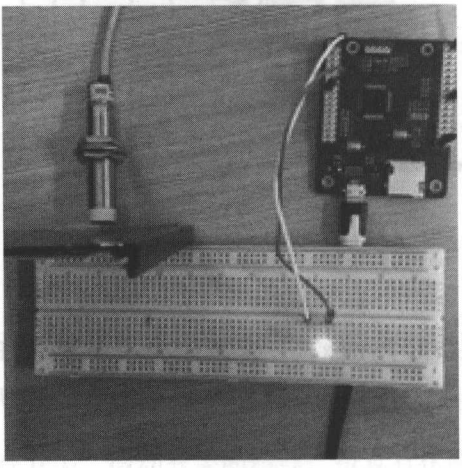


图 10-5　LJ12A3-4-Z/BX 金属接近开关

第 11 章　控制 LCD5110 显示 6×8 字符

在第 4 章，我们学习了点亮心形 8×8 点阵的实验，本章将使用 PyBox 迷你小电脑，通过 LCD 显示屏来显示汉字，向暗恋已久的“女神”表白，祝愿大家早日“脱单”。

零件清单

本章需要的零件如下：

（1）一个 LCD5110 显示屏。

（2）TPYBorad 开发板。

（3）若干条杜邦线。

编写程序代码

在编写程序代码前，需要在 PyBox 上面安装 LCD5110 的驱动，可以从以下地址下载最新的 LCD5110 的驱动：http://www.micropython.net.cn/ueditor/php/

upload/file/20160707/1467854286293558.zip。

下载完成后，把压缩文件解压缩会看到 3 个文件，如图 11-1 所示，其中 font.py 是关于 Python 驱动 LCD5110 显示的字库，upcd8544.py 是驱动 LCD5110 显示的类库，需要把 font.py 和 upcd8544.py 拷贝到 PyBox 的 FLASH 中去，main.py 是测试用例。

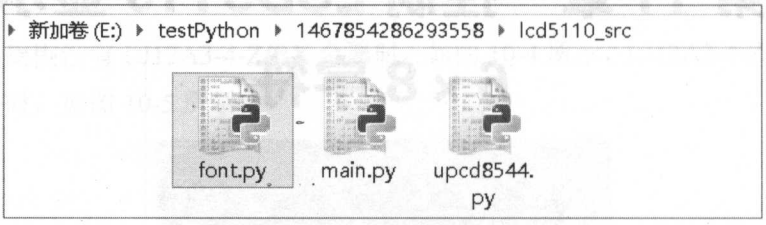


图 11-1　解压缩文件

本案例的相关脚本文件名为：TPY_008.py，需要的具体程序代码如下。

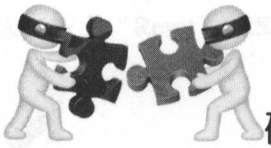
```
import pyb
import upcd8544
from machine import SPI,Pin

def main():
    SPI   = pyb.SPI(1) #DIN=>X8-MOSI/CLK=>X6-SCK
    #DIN =>SPI(1).MOSI 'X8' data flow (Master out, Slave in)
    #CLK =>SPI(1).SCK  'X6' SPI clock

    RST   = pyb.Pin('Y10')
    CE    = pyb.Pin('Y11')
    DC    = pyb.Pin('Y9')
    LIGHT  = pyb.Pin('Y12')
    lcd_5110 = upcd8544.PCD8544(SPI, RST, CE, DC, LIGHT)

    lcd_5110.lcd_write_string('Hello Python!',0,0)
    lcd_5110.lcd_write_string('Micropython',6,1)
    lcd_5110.lcd_write_string('PyBox',12,2)
    lcd_5110.lcd_write_string('v102',60,3)
    lcd_5110.lcd_write_string('This is a test of LCD5110',0,4)

if __name__ == '__main__':
    main()
```



硬件连接

LCD5110 是一种显示屏，通过针脚控制它显示的内容，下面介绍一下 LCD5110 针脚含义(注意：LCD5110 的针脚有些特别)。PyBox 的针脚与 LCD 5110 的针脚对应关系如下：

```
TPYBoard          LCD5110      memo
------------------------------
# any    Pin          => RST          Reset pin (0=reset, 1=normal)
# any    Pin          => CE           Chip Enable (0=listen for input,    1=ignore input)
# any    Pin          => DC           Data/Command (0=commands, 1=data)
#   MOSI              => DIN          data flow (Master out, Slave in)
#   SCK               => CLK          SPI clock
# 3V3    or any Pin   => VCC          3.3V logic voltage (0=off, 1=on)
# any    Pin          => LIGHT        Light (0=on, 1=off)
#   GND               => GND
```

针脚编号如下：

```
TPYBoard    LCD5110   memo
------------------------------
Y10     => RST    Reset pin (0=reset, 1=normal)
Y11     => CE     Chip Enable (0=listen for input, 1=ignore input)
Y9      => DC     Data/Command (0=commands, 1=data)
X8      => DIN    data flow (Master out, Slave in)
X6      => CLK    SPI clock
VCC
Y12     => LIGHT  Light (0=on, 1=off)
GND
```

LCD5110 连线图如图 11-2 所示。

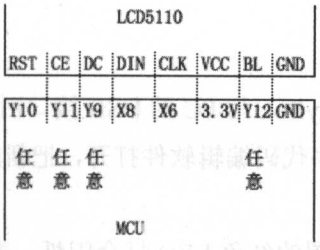


图 11-2　LCD5110 连线图

有图为证

硬件连接图如图 11-3 所示。

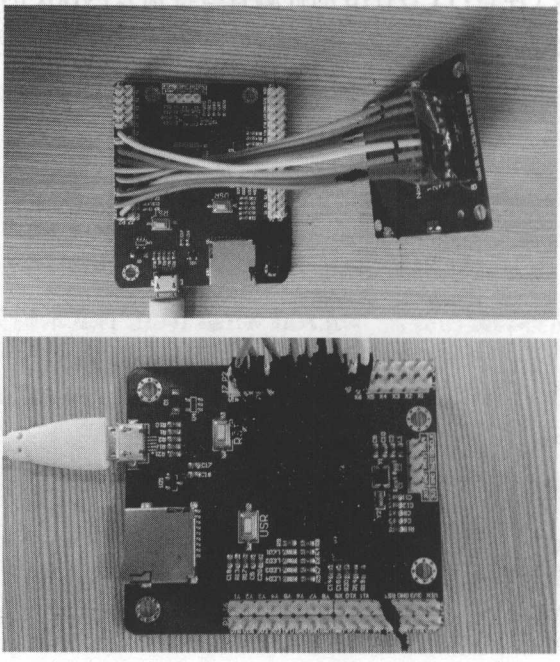


图 11-3　硬件连接图

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件，用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄

灭，按下“Reset”（RST）复位按钮，运行 main.py 文件中的程序代码。

成功时刻

使用 PyBox 迷你小电脑，通过 LCD 显示屏来显示文字的最终效果，如图 11-4 所示。

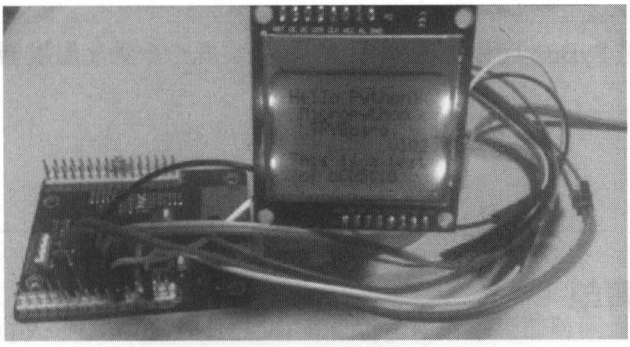


图 11-4　最终效果图

第 12 章　DIY 数字温度计

本章将使用 PyBox 迷你小电脑和 DS3231 实现一个数字温度器，实时显示室温。

零件清单

本章需要的零件如下：

（1）DS3231 模块一个。

（2）PyBox 板子一块。

（3）LCD5110 显示屏一个。

（4）数据线一条。

（5）杜邦线若干条。

编写程序代码

本实验需要 DS3231 时钟模块的驱动程序，从以下地址下载：

http://www.micropython.net.cn/ueditor/php/upload/file/20170111/1484102983654809.rar。

下载后把 DS3231.py 放到 PyBox 的 FLASH 中。

还需要在 PyBox 上面安装 LCD5110 的驱动，从以下地址下载最新的 LCD5110 的驱动：

http://www.micropython.net.cn/ueditor/php/upload/file/20160707/1467854286293558.zip。

下载后把压缩文件解压缩会看到 3 个文件，如图 12-1 所示。其中 font.py 是关于 Python 驱动 LCD5110 显示的字库，upcd8544.py 是驱动 LCD5110 显示的类库，需要把 font.py 和 upcd8544.py 拷贝到 PyBox 的 FLASH 中去。

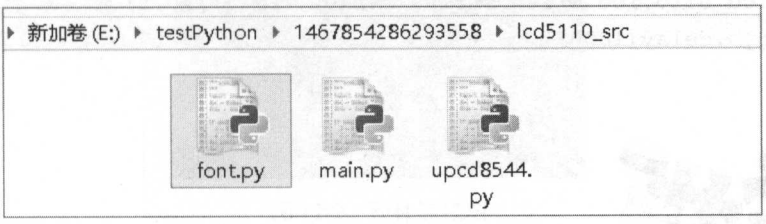


图 12-1　解压缩文件

编写 main.py 文件设定时间，运行 main.py 即可将当前时间显示在 LCD5110 显示屏上。

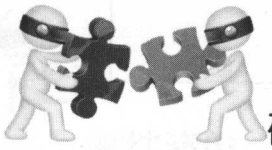
本案例的相关脚本文件名为：TPY_009.py，具体的源代码如下：

```
import pyb
import upcd8544
from machine import SPI,Pin
from DS3231 import DS3231

ds=DS3231(1)
ds.DATE([16,11,26])
ds.TIME([16,14,6])

while True:
    ds.TEMP()
    ds.DATE()
    SPI = pyb.SPI(1) #DIN=>X8-MOSI/CLK=>X6-SCK
    #DIN =>SPI(1).MOSI 'X8' data flow (Master out, Slave in)
    #CLK =>SPI(1).SCK  'X6' SPI clock
```

```
RST    = pyb.Pin('X1')
CE     = pyb.Pin('X2')
DC     = pyb.Pin('X3')
LIGHT  = pyb.Pin('X4')
lcd_5110 = upcd8544.PCD8544(SPI, RST, CE, DC, LIGHT)
lcd_5110.lcd_write_string('Date',0,0)
lcd_5110.lcd_write_string(str(ds.DATE()),0,1)
lcd_5110.lcd_write_string('Time',0,2)
lcd_5110.lcd_write_string(str(ds.TIME()),0,3)
lcd_5110.lcd_write_string('Tem',0,4 )
lcd_5110.lcd_write_string(str(ds.TEMP()),0,5)
pyb.delay(1000)
```



硬件连接

接线好以后，导入 font.py、upcd8544.py 和 DS3231.py 文件，编写 main.py 文件设定时间，运行 main.py 即可将当前时间显示在 LCD5110 显示屏上。

有图为证

硬件连接图如图 12-2 所示。

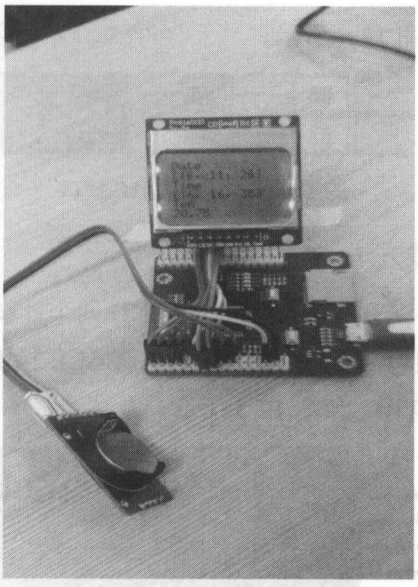
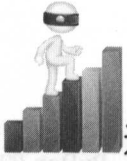


图 12-2　硬件连接图



进阶学习

1．PyBox 的 SPI 接口

LCD5110 需要 SPI 接口与 PyBox 进行连接传输数据，SPI 接口是在 CPU 和外围低速器件之间进行同步串行数据传输，PyBox 有两个 SPI 接口，我们用的是 SPI1 接口。如图 12-3 所示。

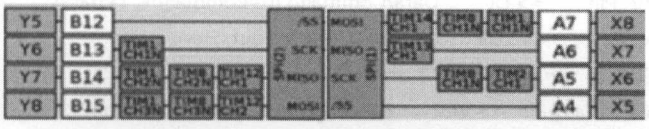


图 12-3　PyBox 的 SPI 接口

2．PyBox 的 I2C 接口

DS3231 是 I2C 接口通信的，通过 I2C 接口与 PyBox 进行数据通信，DS3231 通过这个接口与 PyBox 双向通信，进行数据传输，PyBox 有两个 I2C 接口，我们

用的是 I2C 接口 1。如图 12-4 所示。

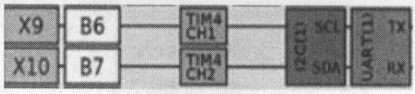


图 12-4　PyBox 的 I2C 接口

3．DS3231 的接线方法

只用到 DS3231 的 SCL、SDA、VCC、GND 四个针脚，即可设定读出当前时间，我们用的是 I2C 接口 1，即 DS3231 的针脚 SCL 接 PyBox 的针脚 *X*9，针脚 SDA 接 PyBox 的针脚 *X*10。如图 12-5 所示。

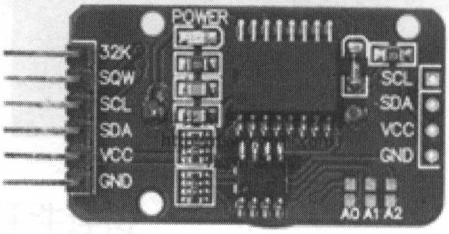


图 12-5　DS3231 的接线方法

4．控制 LCD 5110 显示屏显示 6×8 字符

先看一下 LCD5110 针脚含义（注意：LCD5110 的针脚有些特殊）。PyBox 的针脚与 LCD 5110 的针脚对应关系如下：

```
TPYBoard    LCD5110   memo
-----------------------------
# any  Pin     => RST    Reset pin (0=reset, 1=normal)
# any  Pin     => CE     Chip Enable (0=listen for input,  1=ignore input)
# any  Pin     => DC     Data/Command (0=commands, 1=data)
#  MOSI        => DIN    data flow (Master out, Slave in)
#  SCK         => CLK    SPI clock
# 3V3  or any Pin => VCC    3.3V logic voltage (0=off, 1=on)
# any  Pin     => LIGHT  Light (0=on, 1=off)
#  GND         => GND
```

针脚编号如下：

```
TPYBoard    LCD5110   memo
--------------------------
Y10     => RST    Reset pin (0=reset, 1=normal)
Y11     => CE     Chip Enable (0=listen for input, 1=ignore input)
Y9      => DC     Data/Command (0=commands, 1=data)
X8      => DIN    data flow (Master out, Slave in)
X6      => CLK    SPI clock
VCC
Y12     => LIGHT  Light (0=on, 1=off)
GND
```

连线图如图 12-6 所示。

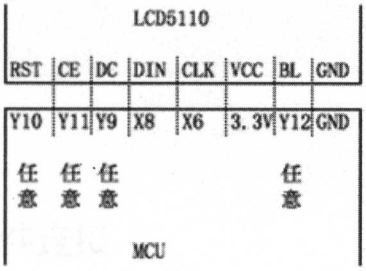


图 12-6　连线图

第 13 章　智能温控小风扇

本章将使用 PyBox 迷你小电脑和零件组装一个智能温控小风扇，为炎炎夏日带来一缕清风。

零件清单

本章需要的零件如下：

（1）PyBox 板子一块。

（2）直流电机一个。

（3）面包板一块。

（4）数据线一条。

（5）三极管 9014（NPN）一个。

（6）杜邦线若干条。

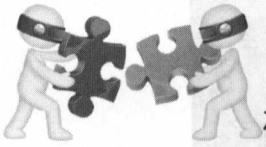
编写程序代码

做智能温控小风扇实验，按照步骤做完以后通电，编写 main.py 文件，即可通过温度控制风扇的转动，本案例的相关脚本文件名为：TPY_010.py，具体代码

如下：

```
import pyb
from pyb import Pin
from ds18x20 import DS18X20

Pin("Y9",Pin.OUT_PP).high()#VCC
Pin("Y11",Pin.OUT_PP).low()#GND
x1 = Pin('X1', Pin.OUT_PP)
pyb.delay(100)
DQ=DS18X20(Pin('Y10'))#DQ
while 1:
    tem = DQ.read_temp()
    if tem > 18:
        x1.value(1)
    else:
        x1.value(0)
```



硬件连接

三极管的原理：在这里我们用到三极管的开关与放大功能，给基极不同的电平控制直流电机电流的通断，以达到控制电机转动的目的，根据三极管特性将集电极连接 PyBox 的 3.3V，发射极连接电机一极，电机另一极接 PyBox 的 GND，通过温度传感器 18B20 检测温度，当温度到达指定温度时，通过 PyBox 控制三极管基极的电平，驱动直流电机转动。如图 13-1 所示。

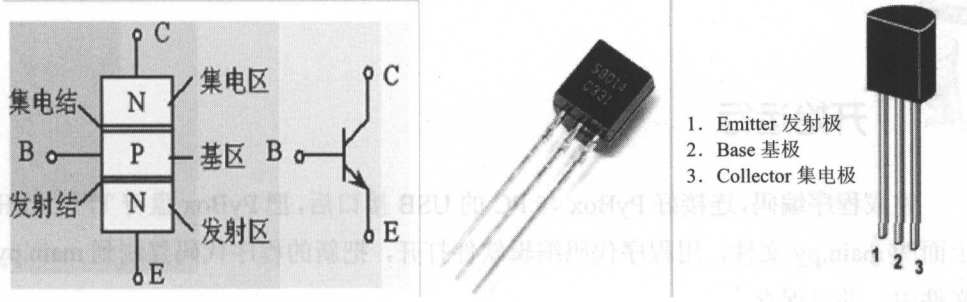


图 13-1　三极管

有图为证

硬件连接图如图 13-2 所示。

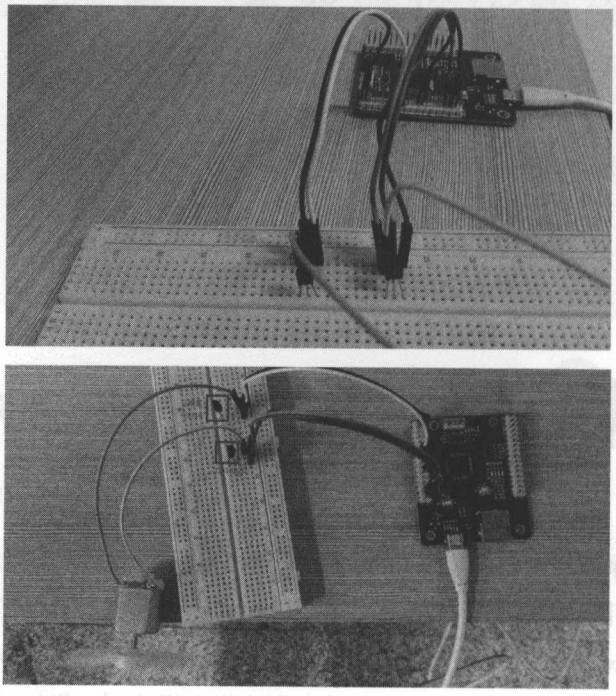


图 13-2　硬件连接图

开始运行

完成程序编码，连接好 PyBox 与 PC 的 USB 接口后，把 PyBox 盘符 TPFLASH 上面的 main.py 文件，用程序代码编辑软件打开，把新的程序代码复制到 main.py 文件中，并且保存。

保存文件时，PyBox 内置的红色 LED 灯会闪烁，表示在写入数据，等红灯熄

灭，按下“Reset”（RST）复位按钮，运行 main.py 文件当中的程序代码。

成功时刻

电扇开始转动了，如图 13-3 所示。

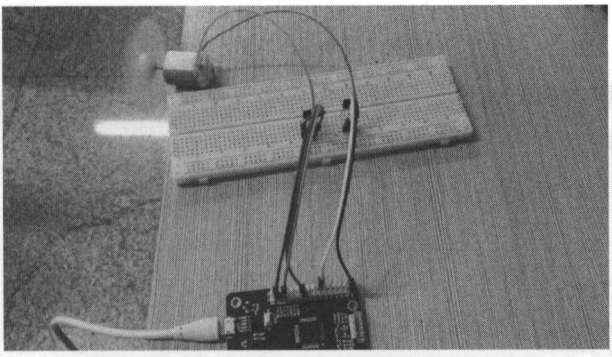


图 13-3　风扇转动

第 14 章　声光电控小夜灯

本章将使用 PyBox 迷你小电脑和零件做一个声光电控小夜灯，在夜晚为勤奋的极客带来光明。

零件清单

本章需要的零件如下：

（1）PyBox 板子一块。

（2）声音传感器一个。

（3）光敏传感器一个。

（4）面包板一块。

（5）发光二极管若干根。

（6）数据线一条。

（7）杜邦线若干条。

编写程序代码

将电源正极、负极与 PyBox 的 3.3V 和 GND 连接起来，然后将光敏传感器与

声音传感器的信号输出针脚连接到 PyBox，笔者的声音传感器信号输出引脚连接的是 PyBox 的 *Y*1 针脚，光敏传感器信号输出引脚连接 TOYBoard 的 *Y*2 针脚，这样传感器就连接完毕。

然后将发光数码管的正极插入面包板正极，负极插入面包板的纵向插孔里（a、b、c、d、e、f、g、h、i、j），用杜邦线将负极连接到 PyBox 的 GND 上，灯的正极连接到 TOYBoard 的 *X*1 针脚，使用声音大小和光亮强度来控制 *X*1 针脚输出高电平或者低电平，以控制发光二极管的亮灭，

接线好以后，编写 main.py 文件，这样我们的 DIY 声光电控开关就完成了。本案例的相关脚本文件名为：TPY_011.py，具体的源代码如下。

```
import pyb
from pyb import Pin

voice = Pin('Y1',Pin.IN)
light = Pin('Y2',Pin.IN)
led = pyb.Pin("X1",pyb.Pin.OUT_PP)

while 1:
    if light.value()==1:
        if voice.value()==1:
            led.value(0)
            pyb.LED(2).off()
            pyb.LED(3).off()
            pyb.LED(4).on()
        else:
            pyb.LED(3).off()
            pyb.LED(4).off()
            led.value(1)
            pyb.LED(2).on()
            pyb.delay(5000)
    else:
        pyb.LED(3).on()
        pyb.LED(2).off()
        pyb.LED(4).off()
        led.value(0)
```

有图为证

硬件连接图如图 14-1 所示。

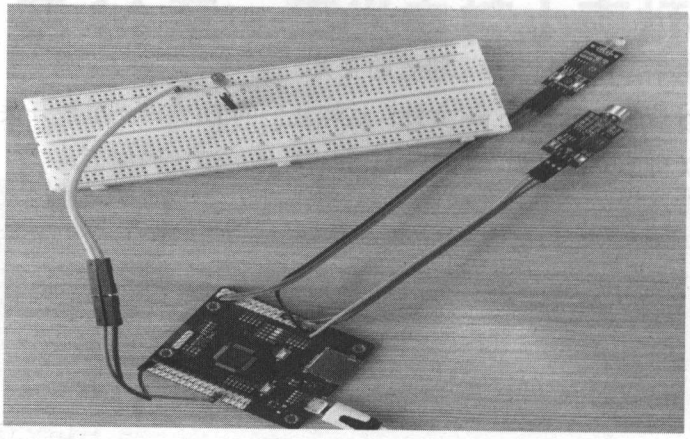


图 14-1　硬件连接图

进阶学习

1. 光敏传感器模块工作原理

（1）光敏电阻模块对环境光线最敏感，一般用来检测周围环境的光线的亮度，触发单片机或继电器模块等。

（2）模块在环境光线亮度达不到设定阈值时，DO 端输出高电平，当外界环境光线亮度超过设定阈值时，DO 端输出低电平。

（3）DO 输出端可以与单片机直接相连，通过单片机来检测高低电平，由此来检测环境的光线亮度。

（4）DO 输出端可以直接驱动继电器模块，由此可以组成一个光控开关。

光敏传感器如图 14-2 所示。

2. 声音传感器模块工作原理

（1）声音模块对环境声音强度最敏感，一般用来检测周围环境的声音强度。

（2）在环境声音强度达不到设定阈值时，模块 OUT 输出高电平，当外界环境声音强度超过设定阈值时，模块 OUT 输出低电平。

（3）小板数字量输出 OUT 可以与单片机直接相连，通过单片机来检测高低电平，由此来检测环境的声音。

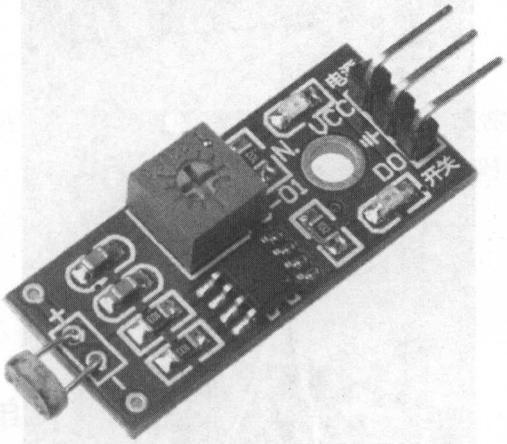


图 14-2　光敏传感器

（4）小板数字量输出 OUT 可以直接驱动继电器模块，由此可以组成一个声控开关。

声音传感器如图 14-3 所示。

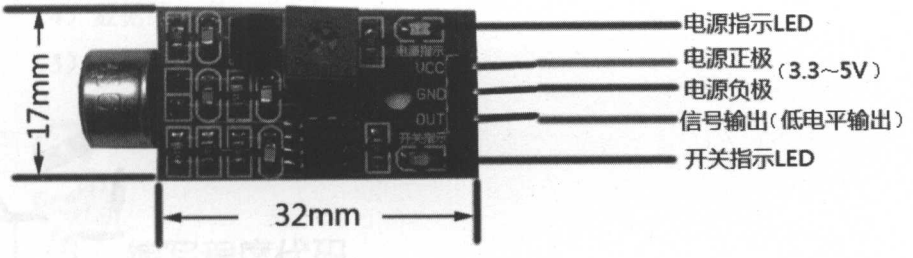


图 14-3　声音传感器

成功时刻

声光电控小夜灯如图 14-4 所示。

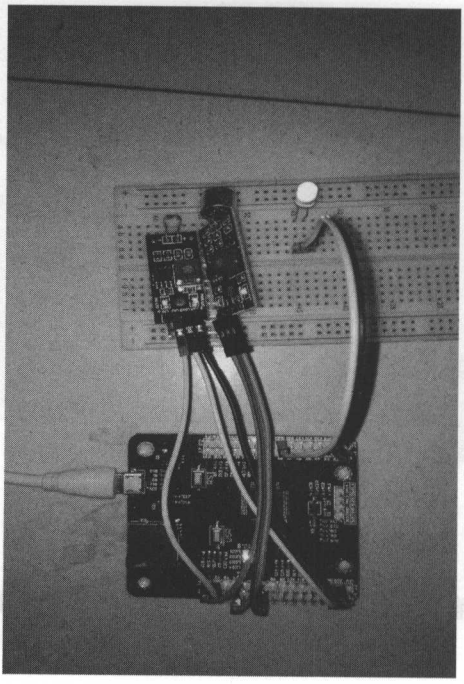


图 14-4　声光电控小夜灯

第 15 章　DIY 超声波测距仪

本章将使用 PyBox 迷你小电脑和零件，做一个超声波测距仪。

超声波测距仪，通过测量从超声波发射到反射回来的时间差，来计算与被测物体之间的距离。可以像蝙蝠一样，实时测量物体距离。

零件清单

本章需要的零件如下：

（1）超声波模块一个。

（2）PyBox 板子一块。

（3）LCD5110 显示屏一个。

（4）数据线一条。

（5）杜邦线若干条。

编写程序代码

编写 main.py 文件，将测到的距离显示在 LCD 5110 显示屏上，运行 main.py 文件。本案例的相关脚本文件名为：TPY_012.py，具体的源代码如下。

```
import pyb
from pyb import Pin
from pyb import Timer
import upcd8544
from machine import SPI,Pin

Trig = Pin('X2',Pin.OUT_PP)
Echo = Pin('X1',Pin.IN)
num=0
flag=0
run=1
def start(t):
    global flag
    global num
    if(flag==0):
        num=0
    else:
        num=num+1
def stop(t):
    global run
    if(run==0):
        run=1
start1=Timer(1,freq=10000,callback=start)
stop1=Timer(4,freq=2,callback=stop)

while True:
    if(run==1):
        SPI = pyb.SPI(1) #DIN=>X8-MOSI/CLK=>X6-SCK
        #DIN =>SPI(1).MOSI 'X8' data flow (Master out, Slave in)
        #CLK =>SPI(1).SCK  'X6' SPI clock
        RST   = pyb.Pin('Y10')
        CE    = pyb.Pin('Y11')
        DC    = pyb.Pin('Y9')
        LIGHT  = pyb.Pin('Y12')
        lcd_5110 = upcd8544.PCD8544(SPI, RST, CE, DC, LIGHT)
        Trig.value(1)
        pyb.udelay(100)
        Trig.value(0)
        while(Echo.value()==0):
            Trig.value(1)
```

```
            pyb.udelay(100)
            Trig.value(0)
            flag=0
        if(Echo.value()==1):
            flag=1
            while(Echo.value()==1):
                flag=1
        if(num!=0):
            #print('num:',num)
            distance=num/10000*34000/2
            print('Distance')
            print(distance,'cm')
            lcd_5110.lcd_write_string('Distance',0,0)
            lcd_5110.lcd_write_string(str(distance),6,1)
            lcd_5110.lcd_write_string('cm',58,1)
            lcd_5110.lcd_write_string('This    is    a    test    of
Distance',0,2)
        flag=0
        run=0
```

有图为证

硬件连接图如图 15-1 所示。

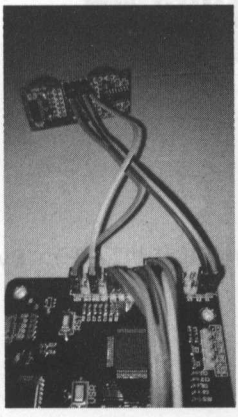


图 15-1　硬件连接图

注意事项

1. 超声波模块工作原理

（1）采用 IO 口 TRIG 触发测距，给最少 10us 的高电平信呈。

（2）模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回。

（3）有信号返回，通过 IO 口 ECHO 输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。测试距离=(高电平时间×声速(340M/S))/2。

接线如图 15-2 所示，VCC 供 5V 电源，GND 为地线，TRIG 触发控制信号输入，ECHO 回响信号输出等 4 个接口端。

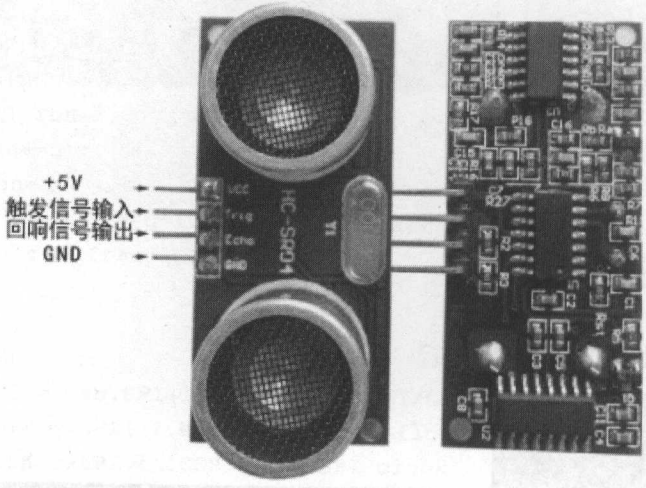


图 15-2　接线图

2. 控制 LCD 5110 显示屏显示 6×8 字符

先看一下 LCD5110 针脚含义（注意：LCD5110 的针脚有些特殊），PyBox 的针脚与 LCD 5110 的针脚对应关系如下：

```
TPYBoard    LCD5110   memo
----------------------------
# any  Pin     => RST    Reset pin (0=reset, 1=normal)
# any  Pin     => CE     Chip Enable (0=listen for input,  1=ignore input)
# any  Pin     => DC     Data/Command (0=commands, 1=data)
#  MOSI        => DIN    data flow (Master out, Slave in)
#  SCK         => CLK    SPI clock
# 3V3  or any Pin => VCC    3.3V logic voltage (0=off, 1=on)
# any  Pin     => LIGHT  Light (0=on, 1=off)
#  GND         => GND
```

针脚编号如下：

```
TPYBoard    LCD5110   memo
-------------------------------------------------------
Y10     => RST    Reset pin (0=reset, 1=normal)
Y11     => CE     Chip Enable (0=listen for input, 1=ignore input)
Y9      => DC     Data/Command (0=commands, 1=data)
X8      => DIN    data flow (Master out, Slave in)
X6      => CLK    SPI clock
VCC
Y12     => LIGHT  Light (0=on, 1=off)
GND
```

连线图如图 15-3 所示。

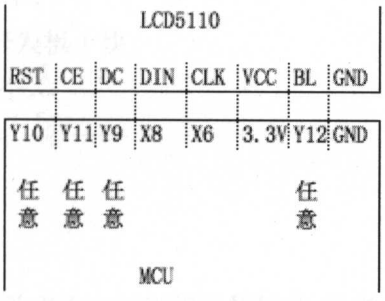


图 15-3　连线图

成功时刻

超声波测距仪最终效果图如图 15-4 所示。

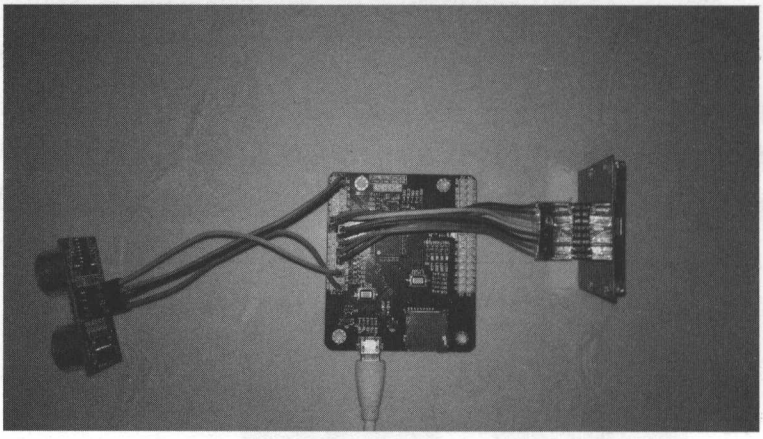


图 15-4　超声波测距仪最终效果图

第 16 章　机器人编程基础——舵机控制实验

《变形金刚》是一部很优秀的电影，相信大家都看过，其中角色分为机器人和汽车人。

虽然现实中还没有变形金刚，不过没关系，本章，我们聪明的 Python 极客，将使用 PyBox 迷你小电脑来控制舵机——机器人的动力基础。

零件清单

本章需要的零件如下：

（1）PyBox V102 开发板 1 块。

（2）SG 90 舵机 1 个。

（3）杜邦线若干条。

实验目的

通过加速度传感器的 X 方向控制舵机的转动，让舵机随 PyBox 的转动而转动。

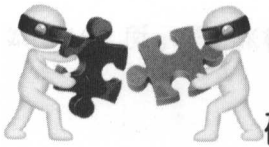
编写程序代码

本案例的相关脚本文件名为：TPY_013.py，具体的程序代码如下。

```
import pyb
import time
from pyb import Pin

xlights = (pyb.LED(2), pyb.LED(3))
ylights = (pyb.LED(1), pyb.LED(4))
M0 = Pin('X1', Pin.OUT_PP)
accel = pyb.Accel()
i=0.0001
j=0.0000
while True:
 x = accel.x()
 print("x=")
 print(x)
 Y=x+20
 M0.high()
 time.sleep(i*Y)
 M0.low()
 time.sleep(i*Y)
 pyb.delay(12)

 if x > 0:
     xlights[0].on()
     xlights[1].off()
 elif x < 0:
     xlights[1].on()
     xlights[0].off()
 else:
     xlights[0].off()
     xlights[1].off()
```



硬件连接

大家应该都看到过机器人的手臂、腿脚等是如何挥动的，它们和机械舞一样有节奏，很多机器人模型里面的动力器件都是舵机。

但是一般见到的动力器件都是像步进电机、直流电机等动力器件，或许大家对舵机比较陌生，但要想使用一个器件，必须要了解它的特性和原理。下面就介绍一下舵机的原理和使用方法。

1．舵机的原理

首先介绍一下原理。先说一下舵机的优点：一是体积紧凑，便于安装；二是输出力矩大，稳定性好；三是控制简单，便于和数字系统接口。本案例使用的是 SG 90 的舵机，笔者感觉性能一般，但是比较稳定，个人做实验使用还是比较不错的。

（1）舵机的组成与参数

舵机又称伺服马达，是一种具有闭环控制系统的机电结构，主要由外壳、电路板、无核心马达、齿轮与位置检测器所构成。其工作原理是由控制器发出 PWM（脉冲宽度调制）信号给舵机，经电路板上的 IC 处理后计算出转动方向，再驱动无核心马达转动，透过减速齿轮将动力传至摆臂，同时由位置检测器（电位器）返回位置信号，判断是否已经到达设定位置，一般舵机只能旋转 180°，有的可以转到 185°。如图 16-1 所示。

（2）舵机的接线

舵机有 3 根线：棕色为地线，红色为电源正极，橙色为信号线，但不同牌子的舵机，相同颜色的线代表意义可能不同。

（3）舵机的控制原理

舵机的转动的角度是通过调节 PWM（脉冲宽度调制）信号的占空比来实现的，标准 PWM（脉冲宽度调制）信号的周期固定为 20ms（50Hz），理论上脉宽分布应在 1～2ms 之间，但事实上脉宽可以在 0.5～2.5ms 之间，脉宽和舵机的转角 0°～180°相对应。

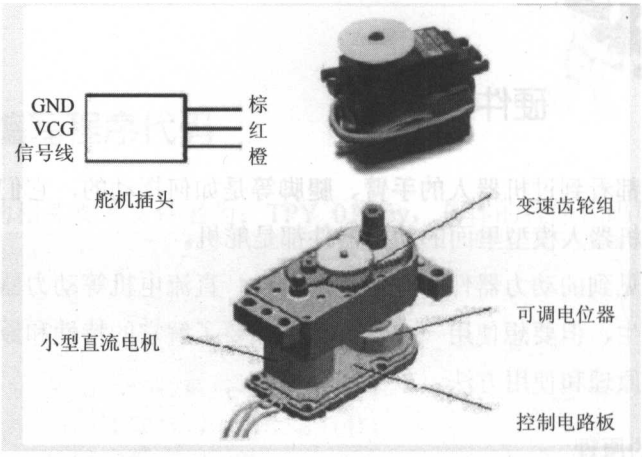


图 16-1　舵机的组成

2. 用 Python 语言控制舵机的方法

以上这些都是为了给大家介绍利用 Python 语言来控制舵机的转动角度，其方法与使用普通单片机一样，都需要用不同宽度的脉冲来控制器转动的角度。我们需要做的就是使用 Python 语言来输出不同宽度的脉冲信号，传输给信号线。现在一般的舵机脉宽在 0.5～2.5ms 之间，这就可以计算了，转 1°的脉冲时间大约是 2 毫秒÷180 度=0.011 毫秒/度，这样依次计算既可。剩下的就是写一个脉冲信号了，相信写脉冲信号的程序大家都很熟悉。

但是，在舵机里面有一些零点几毫秒的延时脉冲，如果在 Python 语言里想要利用 delay()函数来做延时的话，很难做到每个角度都可以转到。所以，需要找到一个比毫秒延时还要精确的延时函数来作为脉冲的计时延时函数。笔者现在用的是 time.sleep(i)这个函数，函数里面的 i 的数值建议设置在 0.0000～0.0035 之间，这样至少 90%的角度都可以转到。如果想让舵机进行循环摆动，一定要记得加上适当的延时，因为程序可以飞快地跑，但是舵机转动也是需要一些时间的，舵机转动时间肯定要比程序跑一遍的时间要长很多。

3. 实验线路图

电路图很简单，只需要给舵机接上 VCC 和 GNG（这是最基本的，开发板上有好多电源和地方可以用），然后把舵机的信号线接到任意一个 GPIO 口就可以了。

如图 16-2 所示是一张模拟的结构原理图。

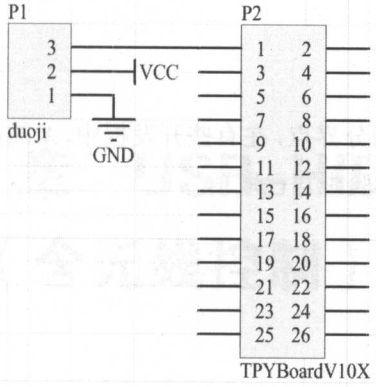


图 16-2　模拟结构原理图

有图为证

硬件连接图如图 16-3 所示。

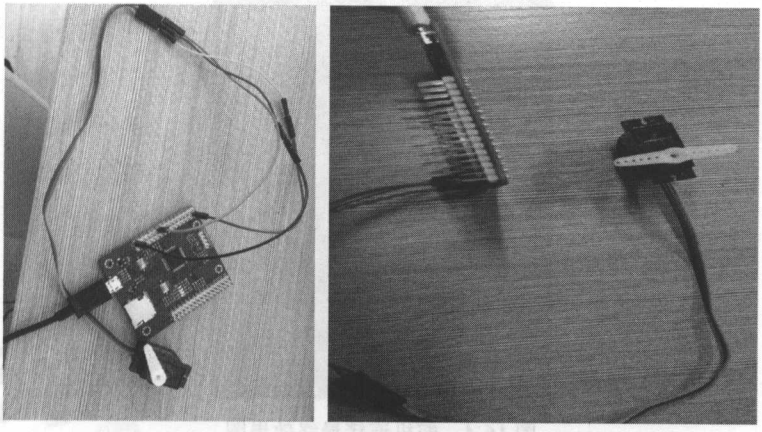


图 16-3　硬件连接图

注意事项

实验中使用 0 做 X 的分界点，在有些开发板中，X 的范围可能在 11～31 之间，所以这个分界点可以在实验中自己取值。

成功时刻

虽然舵机只是能实现转动指定的角度，看起来功能很单一，但是将单一的功能结合起来就能完成很复杂的任务。最终效果图如图 16-4 所示。

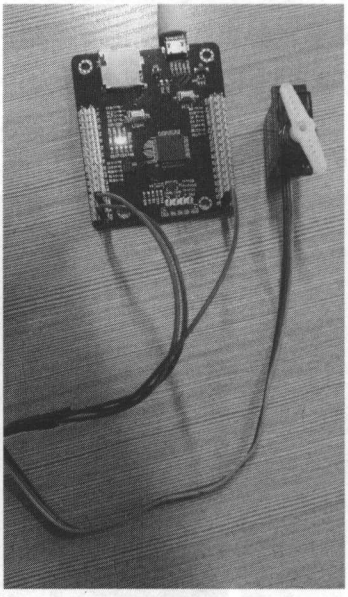


图 16-4　舵机连接最终效果图

第 17 章　USB-HID 测试（含无线控制）

本章笔者将使用 PyBox 迷你小电脑连接 USB-HID 应用，可以用手机摇控键盘输入操作。

零件清单

本章需要的零件如下：

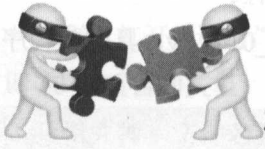
（1）PyBox 板子一块。

（2）数据线一条。

（3）杜邦线若干条。

（4）8×8 LED 点阵一个。

（5）DS18B20 温度传感器一个。



硬件连接

本文以 PyBoxV101 开发板为例，讲解利用 MicroPython 进行 BadUSB 的

USB-HID 设备测试的主要方法，使用 mt7681 模块进行一个简单的实验，实现使用手机摇控键盘输入的测试。

HID 是 Human Interface Device 的缩写，属于人机交互操作的设备，比如 USB 鼠标、USB 键盘、USB 游戏操纵杆、USB 触摸板、USB 轨迹球、电话拨号设备、VCR 遥控等设备。TPYBV101 除了具有 USBHOST 功能以外，还可作为 USB－HID 设备来应用，这里重点讲述如果作为鼠标和键盘使用。

1. PyBoxV101 模拟键盘

该开发板的使用方法可以查看网站 http://www.micropython.net.cn/support_category.php?id=2。在 PyBoxV101 中进行键盘模拟时，每次发送 8 个字符，只要搞清楚这 8 个字符的含义，就能够进行 HID 模拟了。键盘发送的 8 个字符是：BYTE1、BYTE2、BYTE3、BYTE4、BYTE5、BYTE6、BYTE7、BYTE8。其中 BYTE1 用来实现功能键，如图 17-1 所示。

```
BYTE1 --
|--bit0: Left Control 按下时为1
|--bit1: Left Shift按下时为1
|--bit2: Left Alt按下时为1
|--bit3: Left GUI按下时为1
|--bit4: Right Control按下时为1
|--bit5: Right Shift按下时为1
|--bit6: Right Alt按下时为1
|--bit7: Right GUI按下时为1
```

图 17-1　BYTE1 用来实现功能键

BYTE3～BYTE8 是具体按键，比如按下组合键 Left Shift + A，则发送 0x02、0x00、0x04、0x00、0x00、0x00、0x00、0x00。

这里以按下组合键 Left GUI+R 为例，具体讲解实现过程。

第一步：修改 boot.py 文件，相关脚本文件名为：TPY_014a.py，具体的程序代码如下。

```
import machine
import pyb
pyb.usb_mode('CDC+HID',hid=pyb.hid_keyboard)
```

第二步，修改 main.py 文件，相关脚本文件名为：TPY_014b.py，具体的程序

代码如下。

```
hid=pyb.USB_HID()
def release_key_once():
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)
def press_key_once(key):
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = key
    hid.send(buf) # key released
    pyb.delay(10)
def press_2key(key1,key2):
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = key1
    buf[2] = key2
    hid.send(buf) # key released
    pyb.delay(10)
def release_2key():
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = 0
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)

pyb.delay(1000) #开始加入 1 秒延时
press_2key(0x08,0x15)#具体键值见附录部分
release_2key()
```

第三步，安全退出 PyBoxV101，然后按 RST 键，一秒后可以看到“运行”窗口弹出。

2. 简单的 HID 测试

测试打开“运行”窗口，输入 cmd，然后弹出 cmd，输入“shutdown -s -t 60”，即 60 秒后自动关机。

相关脚本文件名为：TPY_014c.py，具体的程序 Main.py 的代码如下。

```
hid=pyb.USB_HID()
def release_key_once():
    buf = bytearray(8) # report is 8 bytes long
```

```
        buf[2] = 0
        hid.send(buf) # key released
        pyb.delay(10)
    def press_key_once(key):
        buf = bytearray(8) # report is 8 bytes long
        buf[2] = key
        hid.send(buf) # key released
        pyb.delay(10)
    def press_2key(key1,key2):
        buf = bytearray(8) # report is 8 bytes long
        buf[0] = key1
        buf[2] = key2
        hid.send(buf) # key released
        pyb.delay(10)
    def release_2key():
        buf = bytearray(8) # report is 8 bytes long
        buf[0] = 0
        buf[2] = 0
        hid.send(buf) # key released
        pyb.delay(10)

    pyb.delay(1000) #开始加入 1 秒延时
    press_2key(0x08,0x15)#具体键值见附录部分
    release_2key()
    pyb.delay(100)
    a=[0x06,0x10,0x07,0x28] #cmd+enter
    for i in a:
        press_key_once(i)
        release_key_once()
    pyb.delay(1000)
    #shutdown -s -t 60 + enter
    a=[0x16,0x0b,0x18,0x17,0x07,0x12,0x1a,0x11,0x2c,0x2d,0x16,0x2c,0
x2d,0x17,0x2c,0x23,0x27,0x28]
    for i in a:
        press_key_once(i)
        release_key_once()
    pyb.delay(1000)
```

程序运行的效果是：当开发板插入电脑后，首先弹出“运行”窗口，然后在

该窗口输入 cmd，此时弹出 cmd，并在其中输入“shutdown -s -t 60”，按回车键，然后电脑在 1 分钟后关机。

3. DIY 一键关机

PyBoxV101 带着一个 USR 按键，可以利用这个按键制作一键关机功能。当开发板程序运行后，按下 USR 按键，产生中断，LED3 闪一下，进行关机操作。相关脚本文件名为：TPY_014d.py，具体的程序代码如下。

```
import pyb
FLAG=0 #flag 标记，当为 1 时，关机

def release_key_once():
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)
def press_key_once(key):
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = key
    hid.send(buf) # key released
    pyb.delay(10)
def press_2key(key1,key2):
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = key1
    buf[2] = key2
    hid.send(buf) # key released
    pyb.delay(10)
def release_2key():
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = 0
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)

def shutdownpc():
    global FLAG
    pyb.LED(3).on()
    FLAG=1
```

```
        pyb.delay(300)
        pyb.LED(3).off()

    hid=pyb.USB_HID()
    sw=pyb.Switch()
    sw.callback(shutdownpc)
    while(1): #led2 闪烁表示板子已经正常工作
        pyb.LED(2).toggle()
        pyb.delay(300)
        print(FLAG)
        if FLAG==1:
            pyb.delay(1000) #开始加入 1 秒延时
            press_2key(0x08,0x15)#具体键值见附录部分
            release_2key()
            pyb.delay(100)
            a=[0x06,0x10,0x07,0x28] #cmd+enter
            for i in a:
                press_key_once(i)
                release_key_once()
            pyb.delay(1000)
            #shutdown -s -t 60 + enter
    a=[0x16,0x0b,0x18,0x17,0x07,0x12,0x1a,0x11,0x2c,0x2d,0x16,0x2c,0
x2d,0x17,0x2c,0x23,0x27,0x28]
            for i in a:
                press_key_once(i)
                release_key_once()
            pyb.delay(1000)
            FLAG=0
```

4. 用手机摇控键盘输入

在这个实验中，笔者使用了 MT7681wifi 模块，该模块可以直接进行串口透传。将 MT7681 与 PyBoxV101 进行连接，接线示意图如图 17-2 所示。这里用的是 PyBoxV101 的 UART3，串口波特率是 115200。

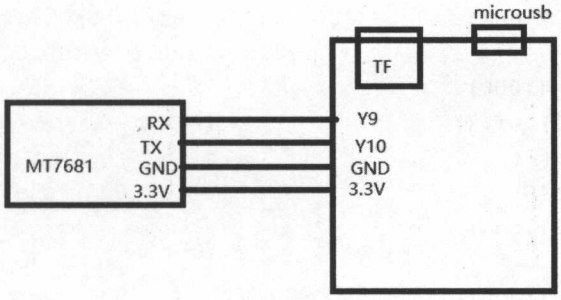


图 17-2　接线示意图

相关脚本文件名为：TPY_014e.py，具体的程序代码如下。

```
import pyb
FLAG=0

def release_key_once():
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)
def press_key_once(key):
    buf = bytearray(8) # report is 8 bytes long
    buf[2] = key
    hid.send(buf) # key released
    pyb.delay(10)
def press_2key(key1,key2):
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = key1
    buf[2] = key2
    hid.send(buf) # key released
    pyb.delay(10)
def release_2key():
    buf = bytearray(8) # report is 8 bytes long
    buf[0] = 0
    buf[2] = 0
    hid.send(buf) # key released
    pyb.delay(10)

def shutdownpc():
    global FLAG
```

```
    pyb.LED(3).on()
    FLAG=1
    pyb.delay(1000)
    pyb.LED(3).off()
def getchars():
    global FLAG
    pyb.LED(3).on()
    FLAG=2
    pyb.delay(1000)
    pyb.LED(3).off()
hid=pyb.USB_HID()
sw=pyb.Switch()
sw.callback(shutdownpc)

u1=pyb.UART(3,115200)
u1.init(115200, bits=8, parity=None, stop=1)
u1.write('Hello world!')
buf=''
#print(buf)
while(1): #led2 闪烁表示板子已经正常工作

    buf=u1.readline()
    print(buf)
    if buf==b's':
        getchars()
    pyb.LED(2).toggle()
    pyb.delay(1300)

    print(FLAG)
    if FLAG==1:
        pyb.delay(1000) #开始加入 1 秒延时
        press_2key(0x08,0x15)#具体键值见附录部分
        release_2key()
        pyb.delay(100)
        a=[0x06,0x10,0x07,0x28] #cmd+enter
        for i in a:
            press_key_once(i)
            release_key_once()
        pyb.delay(1000)
        #shutdown -s -t 60 + enter
```

```
            a=[0x16,0x0b,0x18,0x17,0x07,0x12,0x1a,0x11,0x2c,0x2d,0x16,
0x2c,0x2d,0x17,0x2c,0x23,0x27,0x28]
            for i in a:
                press_key_once(i)
                release_key_once()
            pyb.delay(1000)
            FLAG=0
        if FLAG==2:
            pyb.delay(1000) #开始加入 1 秒延时
            press_2key(0x08,0x15)#具体键值见附录部分
            release_2key()
            pyb.delay(100)
            a=[0x11,0x12,0x17,0x08,0x13,0x04,0x07,0x28] #notepad+enter
            for i in a:
                press_key_once(i)
                release_key_once()
            pyb.delay(1000)
            FLAG=0
```

到这一步，可以看到，手机就像一个摇控器一样，可以直接控制键盘了。只需要在程序中再丰富一下，就能做一个很不错的手机键盘。同时，由于可以通过串口返回数据，所以可以在电脑端编写一个上位机程序，这样就可以把电脑操作的返回值返回来。

注意事项

HID 实际上就是用 PyBox 进行键盘仿真，让我们不再需要键盘，通过 PyBox 就可以完成输入。例如，用于可编程键盘、做可执行 U 盘等设计，甚至被用于一些黑客攻击中。

第三部分　智能小车

第 18 章　无线蓝牙智能小车

每个男孩儿心中都有一个赛车梦，梦想着驾驶酷炫的跑车在赛场上驰骋，本章将使用 PyBox 迷你小电脑和零件来组装一个简单的无线蓝牙智能小车。

零件清单

本章需要的零件如下：

（1）PyBox 板子一块。

（2）蓝牙串口模块一个。

（3）L298N 电机驱动板模块一个。

（4）智能小车底盘一个。

（5）数据线一条。

（6）杜邦线若干条。

编写程序代码

线接好以后，编写 main.py 文件，给 PyBox 通电就完成了，本案例的相关脚本文件名为：TPY_301.py，具体的源代码如下。

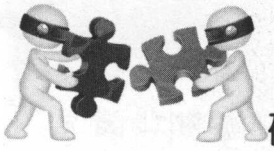
```
import pyb
from pyb import UART
from pyb import Pin

M2 = Pin('X3', Pin.IN)
M3 = Pin('X4', Pin.IN)
N1 = Pin('Y1', Pin.OUT_PP)
N2 = Pin('Y2', Pin.OUT_PP)
N3 = Pin('Y3', Pin.OUT_PP)
N4 = Pin('Y4', Pin.OUT_PP)

u2 = UART(2, 9600)

while True:
    pyb.LED(2).on()
    pyb.LED(3).on()
    pyb.LED(4).on()
    _dataRead=u2.readall()
    if _dataRead!=None:
    #停止（读取手机 APP 传过来的指令，不同的软件指令可能不同，可以自己设定，在这里是默认的，下同）
        if(_dataRead.find(b'\xa5Z\x04\xb1\xb5\xaa')>-1):
            print('stop')
            N1.low()
            N2.low()
            N3.low()
            N4.low()
        #向左
        elif(_dataRead.find( b'\xa5Z\x04\xb4\xb8\xaa')>-1):
            print('left')
            N1.low()
            N2.high()
            N3.high()
            N4.low()
        #向右
        elif(_dataRead.find( b'\xa5Z\x04\xb6\xba\xaa')>-1):
            print('right')
            N1.high()
            N2.low()
            N3.low()
```

```
        N4.high()
    #后退
    elif(_dataRead.find(b'\xa5Z\x04\xb5\xb9\xaa')>-1):
        print('back')
        N2.high()
        N1.low()
        N4.high()
        N3.low()
    #向前
    elif(_dataRead.find( b'\xa5Z\x04\xb2\xb6\xaa')>-1):
        print('go')
        N1.high()
        N2.low()
        N3.high()
        N4.low()
```



硬件连接

1. 蓝牙串口模块原理

（1）引出接口包括 EN、5V、GND、TX、RX、STATE，但小车只用到 RX、TX、GND、5V 四个针脚。

（2）模块默认波特率是 9600，默认配对密码为 1234，默认名称为 HC-06。

（3）LED 指示蓝牙连接状态，闪烁表示没有蓝牙连接，常亮表示蓝牙已连接并打开了端口，当使用安卓系统的手机软件发送指令时，通过串口给 PyBox 发送指令，PyBox 收到指令通过 L298BN 模块来驱动小车前进、后退、向左、向右或者停止。

如图 18-1 所示的接线，5V 接 PyBox 的 VIN，GND 为地线，TX 接 PyBox 的 RX（这里用的是 PyBox 串口 2、*X*3、*X*4），即 *X*4，RX 接 PyBox 的 TX，即 *X*3。

2. 学习 L298N 电机驱动板模块的接线方法

本模块是两路的 H 桥驱动，可以同时驱动两个电机，方法如图 18-2 所示，使得 ENA、ENB 两个插脚可以分别从 IN1、IN2 输入 PWM 信号驱动电机 1 的转速

和方向，还可以分别从 IN3、IN4 输入 PWM 信号驱动电机 2 的转速和方向。我们将电机 1 接口的 OUT1、OUT2 与小车的一个电机的正负极连接起来，将电机 2 接口的 OUT3、OUT2 与小车的另一个电机的正负极连接起来。然后将两边的接线端子，即供电正极（中间的接线端子为接地）连接 PyBox 的 VIN，中间的接线端子即接地，连接 PyBox 的 GND，In1-In4 连接 PyBox 的 *Y*1、*Y*2、*Y*3、*Y*4，通过 *Y*1、*Y*2、*Y*3、*Y*4 的高低电平，来控制电机的转动，从而让小车前进、后退、向左、向右。

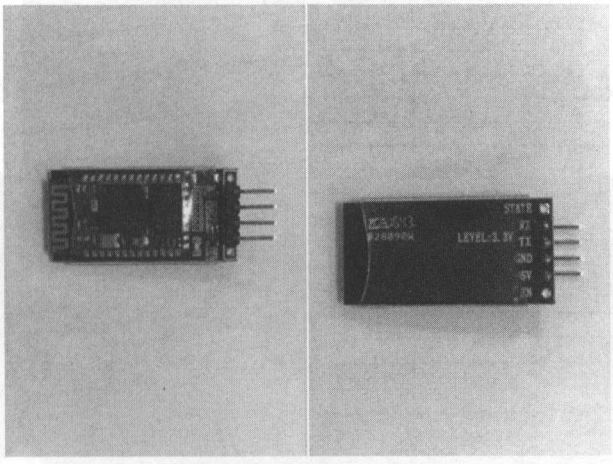


图 18-1　接线

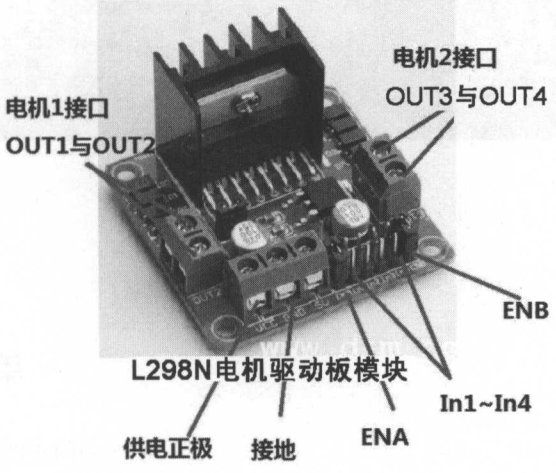


图 18-2　L298N 电机驱动板模块的接线

有图为证

硬件连接图如图 18-3 所示。

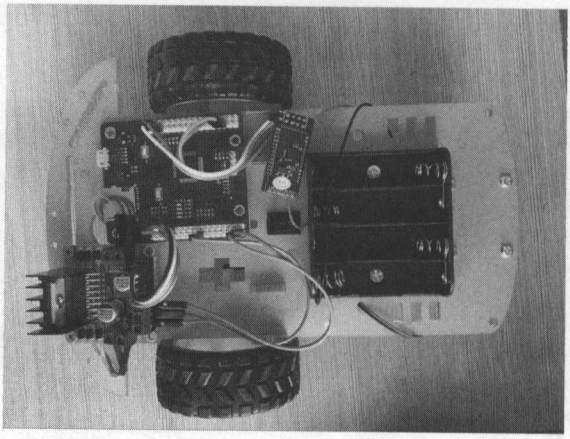


图 18-3　硬件连接图

第 19 章　红外寻迹无线小车

本章将带领 Python 极客们更进一步地实现赛车梦，使用 PyBox 开发板制作一个红外寻迹小车，如图 19-1 所示。

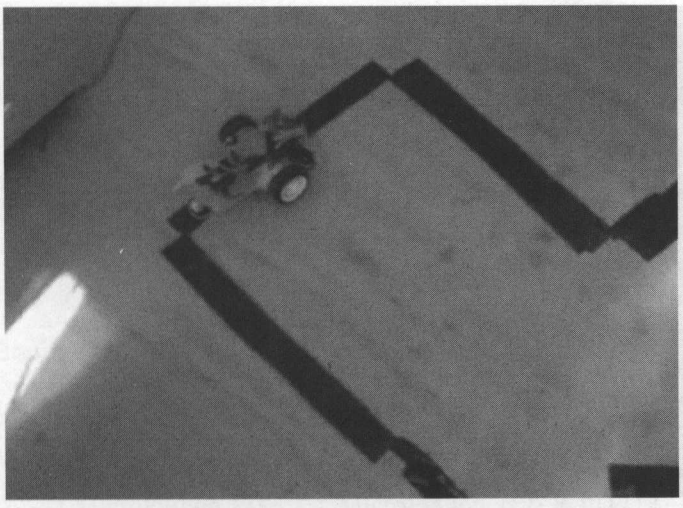


图 19-1　红外寻迹小车

零件清单

本章需要的零件如下：

（1）PyBox 开发板 1 块（能运行 Python 语言的开发板，即小车的大脑）。

（2）四路红外感应探头（即小车的眼睛）。

（3）数据线一根。

（4）充电宝一个（给整个系统供电）。

（5）智能小车底盘（包括驱动模块）。

（6）杜邦线若干条。

编写程序代码

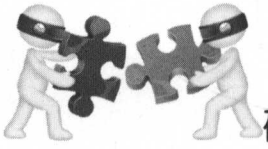
本案例的相关脚本文件名为：TPY_302.py，具体的源代码如下。

```
import pyb
from pyb import UART
from pyb import Pin

M0 = Pin('X1', Pin.IN)
M1 = Pin('X2', Pin.IN)
M2 = Pin('X3', Pin.IN)
M3 = Pin('X4', Pin.IN)
N1 = Pin('Y1', Pin.OUT_PP)
N2 = Pin('Y2', Pin.OUT_PP)
N3 = Pin('Y3', Pin.OUT_PP)
N4 = Pin('Y4', Pin.OUT_PP)

print('while')
while True:
    print('while')
    pyb.LED(4).off()
    pyb.LED(3).off()
    pyb.LED(2).off()
    if(M0.value()==1):#检测到物体返回 0。
        pyb.LED(4).on()
        pyb.delay(50)
        N1.low()
        N2.high()
        N3.low()
```

```
        N4.high()
        pyb.delay(30)
        #pyb.delay(5000)
    if(M3.value()==1):#检测到物体返回 0。
        pyb.LED(3).on()
        pyb.delay(50)
        N1.high()
        N2.low()
        N3.high()
        N4.low()
        pyb.delay(30)
    if(M2.value()&M1.value()==1):
        pyb.LED(2).on()
        N1.low()
        N2.high()
        N3.high()
        N4.low()
        pyb.delay(70)
```



硬件连接

1. PYBoard 开发板

MicroPython 是在单片机上可以运行的 Python，也就是说，可以通过 Python 脚本语言开发单片机程序。由剑桥大学的理论物理学家乔治·达明设计，与 Arduino 类似，但 MicroPython 更强大。MicroPython 开发板可以通过 Python 代码轻松控制微控制器的各种外设，比如 LED 灯、读取管脚电压、播放歌曲、与其他设备联网等。PyBox 是 TurnipSmart 公司制作的一款 MicroPython 开发板，这款开发板运行很流畅，而且价格很便宜。

2. 四路红外感应探头

（1）当模块检测到前方障碍物的信号时，电路板上红色指示灯将被点亮，同时 OUT 端口持续输出低电平信号，该模块检测距离 2～60cm，检测角度是 35°，检测距离可以通过电位器进行调节：顺时针调电位器，检测距离增加；逆时针调

电位器，检测距离减少。

（2）传感器属于红外线反射探测，因此目标的反射率和形状是探测距离的关键。其中黑色探测距离最小，白色最大；小面积物体距离小，大面积物体的距离大。

（3）传感器模块输出端口 OUT 可直接与单片机 IO 口连接，也可以直接驱动一个 5V 继电器模块或者蜂鸣器模块，连接方式：VCC-VCC;GND-GND;OUT-IO。

（4）比较器采用 LM339，工作稳定。

（5）可采用 3.3～5V 直流电源对模块进行供电。当电源接通时，绿色电源指示灯将被点亮。

3. 智能小车底盘

双电机驱动，万向轮改变方向。这是实验中最常用到的小车底盘，使用差速的方法进行转弯。配合使用 L298N 电机驱动模块，使用方法很简单。

4. 寻迹原理

下面说一下小车寻迹的原理。

（1）红外探头的安装

小车寻迹的原理其实就是光的吸收、反射和散射。大家都知道，白色反射所有颜色的光，而黑色吸收所有颜色的光，这就为小车寻迹提供了有力的科学依据。在小车的车头上安装红外探头，按一字顺序排开。哪个探头接收不到反射或者散射回来的光时，说明这个探头此时正在黑色的寻迹带上。

（2）返回信号的判断

如果要是正前方的探头接收不到光，那么说明小车此时走在黑色的寻迹带上。可以使小车直线行走。如果左面的探头接收不到光，那么说明小车左面出现了黑色寻迹带，此时小车应该执行左转弯。右转弯与左转弯原理相同。

如果要是小车前面、左面、右面三个方向全都接收不到光，或者是两个方向上的探头都接收不到光，那么到底是左转弯、右转弯还是继续直行？这个就要看在程序里是怎么做判断的了。

（3）硬件接线

四路红外探头接线很简单，虽然有 18 根线，但是有 12 根线是每 3 根组成一组，分成四组，互相对应，操作很简单，还有 4 根线直接接到单片结 IO 口上，剩下的 2 根接到 VCC 和 GND。

硬件连接图如图 19-2 所示。

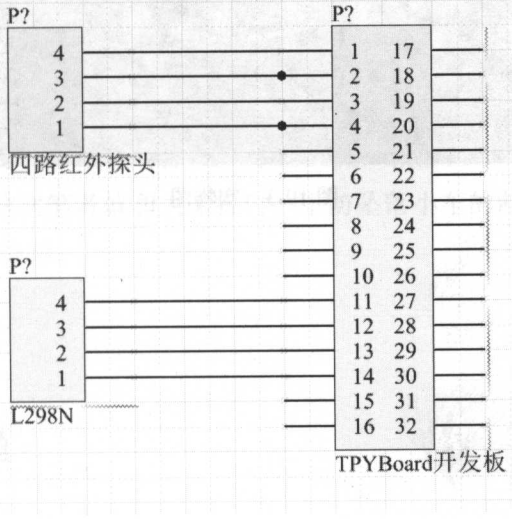


图 19-2　硬件连接图

5．运行与调试

制作完成后，剩下的就是调试了，调试中应该注意细节和小车稳定性的优化。

有图为证

实物图如图 19-3 所示。

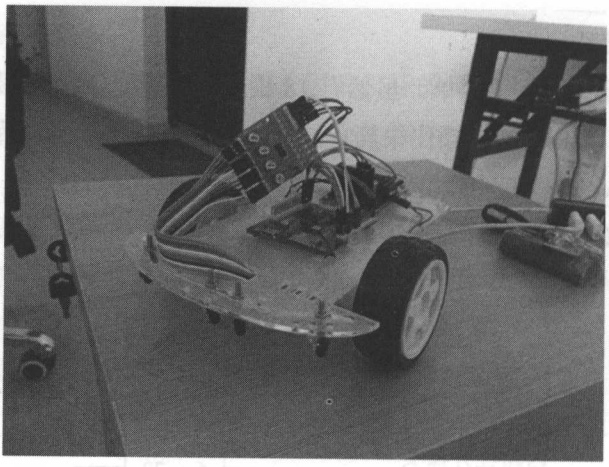


图 19-3　实物图

第 20 章　红外防坠落小车

现在智能小车是电子竞赛或者 DIY 中的主流，相信大家对寻迹、壁障、遥控也都见的很多了，这次笔者就和大家探讨一下防坠落小车的制作方法，本次的程序使用 Python 语言编写。

零件清单

本章需要的零件如下：

（1）PyBox 开发板 1 块（能跑 Python 语言的开发板，即小车的大脑）。

（2）四路红外感应探头（小车的眼睛）。

（3）数据线一根。

（4）充电宝一个（给整个系统供电）。

（5）智能小车底盘（包括驱动模块）。

（6）杜邦线若干条。

编写程序代码

本案例的相关脚本文件名为：TPY_303.py，本案例使用的源代码如下：

```
import pyb
from pyb import UART
from pyb import Pin

M0 = Pin('X1', Pin.IN)
M1 = Pin('X2', Pin.IN)
M2 = Pin('X3', Pin.IN)
M3 = Pin('X4', Pin.IN)
N1 = Pin('Y1', Pin.OUT_PP)
N2 = Pin('Y2', Pin.OUT_PP)
N3 = Pin('Y3', Pin.OUT_PP)
N4 = Pin('Y4', Pin.OUT_PP)

print('while')
while True:
    print('while')
    if(M2.value()|M1.value()|M3.value()|M0.value()==0):
        N1.low()
        N2.high()
        N4.high()
        N3.low()
        pyb.LED(2).on()
        pyb.LED(3).off()
    elif(M2.value()|M1.value()|M3.value()|M0.value()==1):
        N1.high()
        N2.low()
        N4.low()
        N3.high()
        pyb.delay(300)
        N1.low()
        N2.high()
        N3.high()
        N4.low()
        pyb.delay(200)
        pyb.LED(3).on()
        pyb.LED(2).off()
```

这个案例的原理与红外寻迹小车类似，有关的介绍我们不再重复。

需要注意的是，这个案例的不同之处在于：

小车在行驶路面检测到物体的时候，说明前面是有路的，不是悬空的。那么

小车保持直行。如果检测到前方没有返回，说明没有检测到物体，也说明前面没有路，是悬空的，那么小车就先进行后退，再进行右转（或者也可以左转，也可以一次右转一次左转，这个比较随意）。

4．硬件接线

接线其实很简单，四路红外探头接线，虽然有 18 根线，但是有 12 根是每 3 根分成四组的，有 2 根接在 VCC 和 GND，还有 4 根直接接到单片结 IO 口上。L298N 的接线更简单了，这里不多介绍。

如图 20-1 所示是一个简单的帮助理解的原理图。

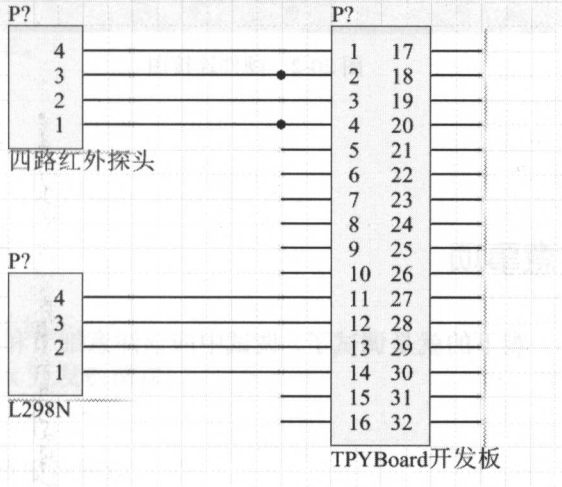


图 20-1　连线图和原理图

有图为证

如图 20-2 所示为硬件连接实物图。

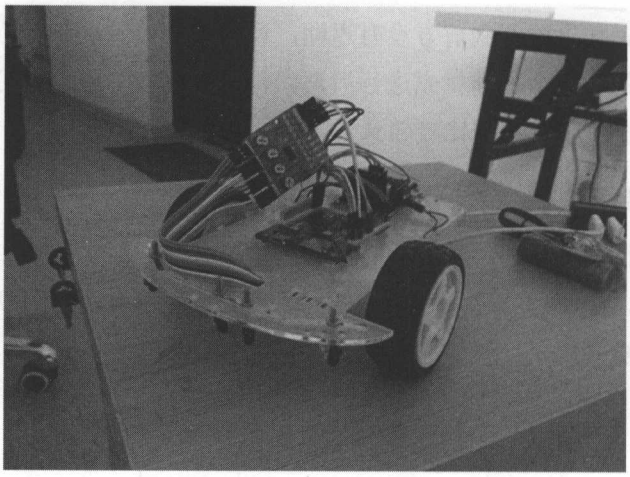


图 20-2　硬件连接图

注意事项

制作完成后，剩下的就是调试了，调试中应该注意细节和小车稳定性的优化。

第 21 章　加速度传感器无线小车

现在无线控制已经成为了电子科学领域的主流，这次笔者就教大家做一个主流中的无线控制小车。

零件清单

本章需要的零件如下：

（1）PyBoxV10x 开发板两块。

（2）小车底盘一个。

（3）LORA 无线模块两块。

（4）充电宝一个。

（5）9014 三极管两个。

编写程序代码

控制端相关脚本文件名为：TPY_304a.py，具体源代码如下。

```
import pyb
```

```
xlights = (pyb.LED(2), pyb.LED(3))
ylights = (pyb.LED(1), pyb.LED(4))

from pyb import UART
from pyb import Pin
#from ubinascii import hexlify
from ubinascii import *

accel = pyb.Accel()
u2 = UART(2, 9600)
i=0
K=1
```

控制端主程序相关脚本文件名为：TPY_304b.py，具体源代码如下。

```
  print('while')
while (K>0):
    _dataRead=u2.readall()
    if(1>0):
        x = accel.x()
        print("x=")
        print(x)
        if (x > 10):
            xlights[0].on()
            xlights[1].off()
            u2.write('\x00\x05\x18YOU')
            #pyb.delay(1000)
            print('\x00\x01\x18YOU')
        elif (x < -10):
            xlights[1].on()
            xlights[0].off()
            u2.write('\x00\x05\x18ZUO')
            print('\x00\x01\x18ZUO')
            #pyb.delay(1000)

        else:
            xlights[0].off()
            xlights[1].off()
```

```
        y = accel.y()
        print("y=")
        print(y)
        if (y > 15):
            ylights[0].on()
            ylights[1].off()
            #u2.write('\x00\x05\x18HOU')
            #pyb.delay(1000)
            #print('\x00\x01\x18HOU')
        elif (y < -15):
            ylights[1].on()
            ylights[0].off()
            u2.write('\x00\x05\x18QIAN')
            #pyb.delay(1000)
            print('\x00\x01\x18QIAN')
        else:
            ylights[0].off()
            ylights[1].off()

        pyb.delay(10)
```

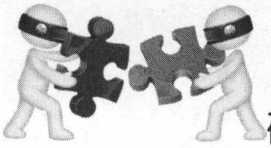
被控制端源代码相关脚本文件名为：TPY_304c.py，具体代码如下。

```
import pyb
from pyb import UART
from pyb import Pin
from ubinascii import hexlify
from ubinascii import *
M1 = Pin('X1', Pin.OUT_PP)
M3 = Pin('Y1', Pin.OUT_PP)
u2 = UART(2, 9600)
i=0
K=1
```

被控制端主程序相关脚本文件名为：TPY_304d.py，具体代码如下。

```
print('while')
while (K>0):
    M1.high()
    pyb.delay(3)
    M3.high()
```

```
    if(u2.any()>0):
            print('1234')
            M1.low()
            M3.low()
            pyb.delay(3)
            _dataRead=u2.readall()
            print('123',_dataRead)
            if(_dataRead.find(b'QIAN')>-1):
                M1.low()
                M3.low()
                print('QIAN')
                pyb.delay(250)
            elif(_dataRead.find(b'ZUO')>-1):
                M1.low()
                M3.high()
                print('ZUO')
                pyb.delay(250)
            elif(_dataRead.find(b'YOU')>-1):
                M1.high()
                M3.low()
                print('ZUO')
                pyb.delay(250)
```



硬件连接

首先介绍一下加速度传感器。

加速度传感器工作原理是：主要利于压电敏感元件的压电效应得到与振动或者压力成正比的电荷量或者电压量。目前工业现场采用典型 IEPE 型加速度传感器，及内置 IC 电路压电加速度传感器，传感器输出是与振动量成正比的电压信号，例如：100mV/g（每个加速度单位输出 100mV 电压值。1g=9.81m/s-2）。

如图 21-1 所示为控制端原理图，图 21-2 所示为被控制端原理图。

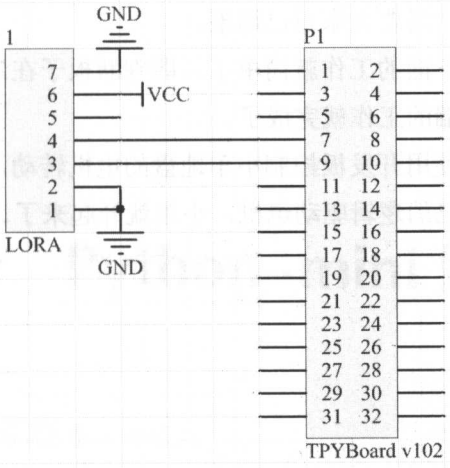


图 21-1　控制端原理图

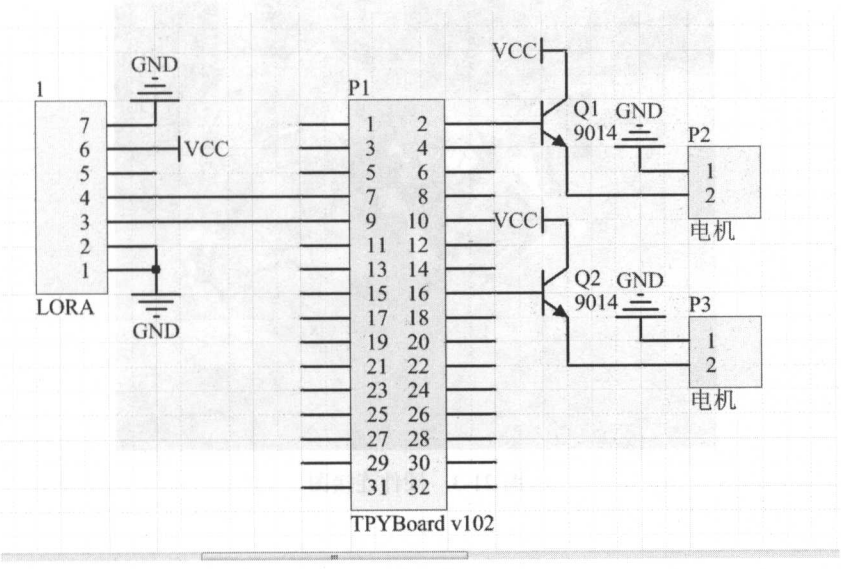


图 21-2　被制端原理图

以上两个原理图还是挺复杂的，不过没关系，大家知道大概的思路就可以了。

简单来说，这里的控制端主要指加速度传感器，通过测量由于重力引起的加速度，计算出设备相对于水平面的倾斜角度、设备移动的方式。

直接使用 MicroPython 语言里相关的函数就行。

需要注意的是，函数返回的倾斜度数值，因为各个传感器品牌不同、做工差

异，所以返回值不同，需要大家自己调整。

得到倾斜值后，下面的工作就简单了，即判断板子在怎么倾斜，然后把倾斜的信号传出去，控制端的工作就完成了。

被控制端，就是使用开发板控制小车地盘的电机转动，根据收到的数据做判断。判断后再按照自己的逻辑驱动电机，小车就开起来了。

有图为证

硬件连接图如图 21-3 所示。

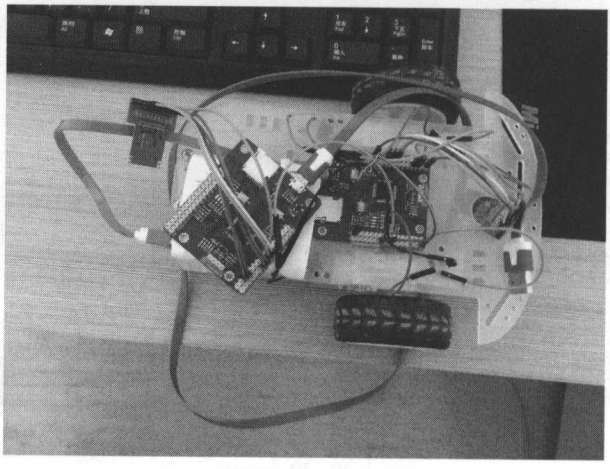
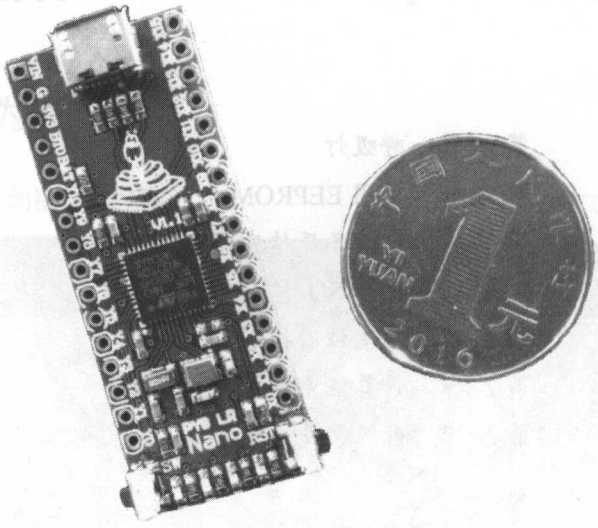


图 21-3　硬件连接图

第四部分　Python-mini 编程案例

Python -mini 主板说明

Python-mini 主板是 MicroPython 中文社区为了推广、宣传 MicroPython 而设计和制作，它是目前体积最小、成本最低、兼容官方 PYBoard 的 MicroPython 开发板，非常适合用于 MicroPython 入门。和官方 PYBoard 一样，Python-mini 也是完全开源的。



Python-mini 的主要特点：

- 带有 microUSB，支持 USB 升级功能。
- 2 路 UART、3 路 I2C、3 路 SPI。
- 10 路 12 位 ADC。
- 带有 RTC，支持后备电池输入。
- 支持 USB 供电和 VIN 输入（最高 12V）。
- 带有一个用户按键和一个复位键。
- 带有 4 个支持亮度调节功能的 LED。
- 带有加速度传感器（MMA7660）。
- 丰富的参考应用。
- 低成本、高性能。
- 完全开源。

应用范围：

- 教育、学习
- 电子竞赛
- 机器人
- 智能硬件
- 物联网开发
- 快速原型设计
- 创客、DIYer

第 22 章　呼吸灯

本案例的相关脚本文件名为：NPY_001.py，下面程序将 LED（3）设置为呼吸灯。

```
from pyb import Timer

ia = 1
da = 1
def fa(t):
 global ia, da
 if (ia==0)or(ia==255):
         da=256-da
     ia=(ia+da)%256
 pyb.LED(3).intensity(ia)

tm=Timer(1, freq=200, callback=fa)
```

如果将 LED（3）改为其他数字（范围是 1～4），那么可以将其他 LED 变为呼吸灯。

如果将这段程序复制到 main.py 中，那么每次上电就会自动显示呼吸灯。

1．RTC 时钟

PYB mini 连接 DS3231 时钟模块，接线方式如表 22-1 所示。连线图如图 22-1 所示。

表 22-1　接线方式

DS3231	PYB mini
GND	GND
VCC	3V3
SDA	Y0/PB9
SCL	Y1/PB8

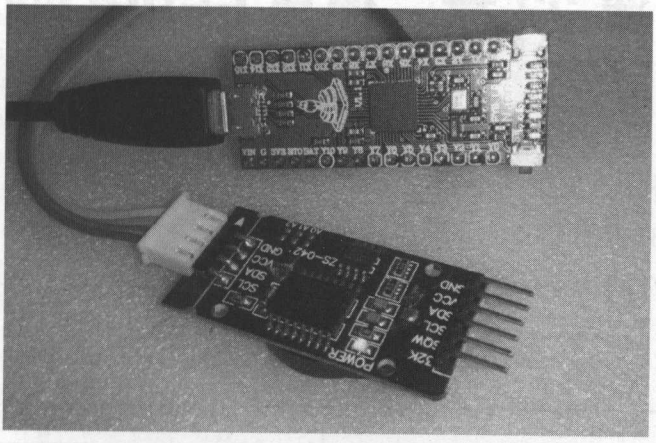


图 22-1　连线图

将 DS3231 库导入，就可以直接使用了。

```
>>>from DS3231 import DS3231
>>> ds=DS3231(1)
>>> ds.sec() #读取秒
46
>>> ds.sec()
47
>>> ds.TIME()    #读取时间，时：分：秒
[22, 35, 56]
>>> ds.DATE()    #读取日期，年-月-日
[16, 10, 20]
>>> ds.TEMP()    #读取 DS3231 的温度传感器
26.5
>>>
```

如果每秒读取一次 RTC，就可以看到时间的变化了。

第 23 章　使用 EEPROM

DS3231 模块上还有一个 EEPROM AT24C32 保存数据，它也是 I2C 接口的应用，与 DS3231 共用 I2C。

其接线方式和连线图与 DS3231 一样，接线方式如表 23-1 所示，连线图如图 23-1 所示。

表 23-1　接线方式

DS3231	PYB mini
GND	GND
VCC	3V3
SDA	Y0/PB9
SCL	Y1/PB8

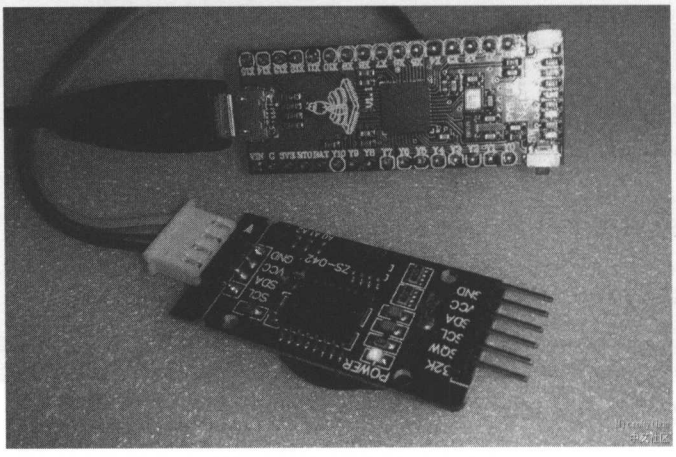


图 23-1　连线图

本案例的相关脚本文件名为：NPY_002.py，具体的源代码如下。

```
frompybimportI2C

_24L64_ADDR = const(0x57)

class _24L64(object):
def __init__(self, i2c_num, i2c_addr=_24L64_ADDR, i2c_baud=100000):
        self.i2c_addr = i2c_addr
        self.i2c_buad = i2c_baud
        self.r = bytearray(2)
        self.w = bytearray(3)
        self.i2c = I2C(i2c_num, I2C.MASTER, baudrate = i2c_baud)

def read(self, addr):
        self.r[0] = addr//256
        self.r[1] = addr%256
        self.i2c.send(self.r, self.i2c_addr)
return self.i2c.recv(1, self.i2c_addr)[0]

def write(self, addr, dat):
        self.w[0] = addr//256
        self.w[1] = addr%256
        self.w[2] = dat
        self.i2c.send(self.w, self.i2c_addr)
```

运行效果：

```
>>>from _24L64 import _24L64
>>> ee=_24L64(1)
>>> ee.read(0)
0
>>> ee.read(1)
2
>>> ee.write(1, 5)
>>> ee.read(1)
5
```

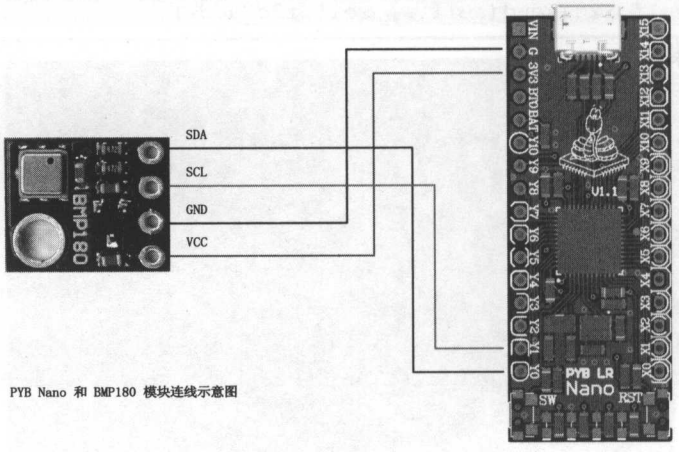
第 24 章　使用气压传感器 BMP180

BMP180 也是常用的气压传感器，它的特点是体积小、接口简单（I2C）。下面在 PYB mini 上演示使用 BMP180。

接线方式如表 24-1 所示，如图 24-1 所示。

表 24-1　接线方式

BMP180	PYB mini
GND	GND
VCC	3V3
SDA	Y0/PB9
SCL	Y1/PB8



PYB Nano 和 BMP180 模块连线示意图

图 24-1　连线图（1）

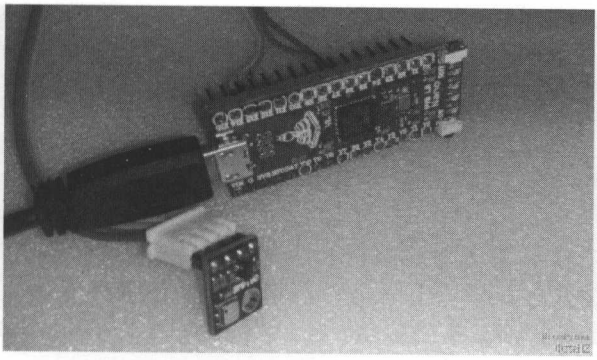


图 24-1　连线图（2）

然后导入 BMP180 的库，就可以使用了。

```
from bmp180 import BMP180
b=BMP180(1)
b.getPress()
```

主要的函数有：

- b.getTemp()，获取温度。
- b.getPress()，获取气压。
- b.getAltitude()，获取高度。
- b.get()，获取全部参数。

1. 磁场传感器 HMC5883 的使用

HMC5883 是一个 I2C 接口的磁传感器芯片，应用于低成本罗盘和磁场检测。

因为 HMC5883 模块上带有 LDO，所以它的 VCC 接到了 PYB mini 的 VIN 上（4.8V 左右）。接线方式如表 24-2 所示，连线图如图 24-2 所示。

表 24-2　接线方式

HMC5883	PYB mini
GND	GND
VCC	3V3
SDA	Y0/PB9
SCL	Y1/PB8

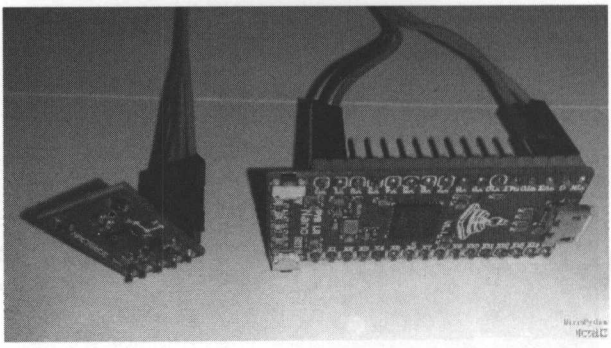


图 24-2　连线方式图

本案例的相关脚本文件名为：NPY_003.py，具体的程序代码如下。

```
>>>from HMC5883 import HMC5883
>>> h = HMC5883(1)
>>>while True:
...     pyb.delay(500)
...     print("%5d"%h.getX(),"%5d"%h.getY(),"%5d"%h.getZ())
...
65387   407    73
65384   406    74
65386   403    78
65386   403    78
65383   404    77
65384   406    76
65385   404    77
   56   402 65215
  124   351 65162
  117   341 65161
   44   313 65110
  149   375 65362
   60   405 65528
65490   413    49
65424   421    28
```

第 25 章　使用 SD 卡

因为引脚数量的限制，所以在 STM32F401CEU6 上没有 SD 接口，因此 PYB mini 上也没有 SD/microSD，这给存储大数据带来不便。不过我们可以通过 SPI 接口挂载 SD，下面介绍具体的连接方法。

通过 SPI 方式连接需要 6 根线（包括电源）。接线方式如表 25-1 所示，连线图如图 25-1 所示。

表 25-1　接线方式

DS3231	PYB mini
GND	GND
VCC	3V3
MOSI	X14/PB15
MISO	X13/PB14
SCK	X12/PB13
CS	X11/PB12

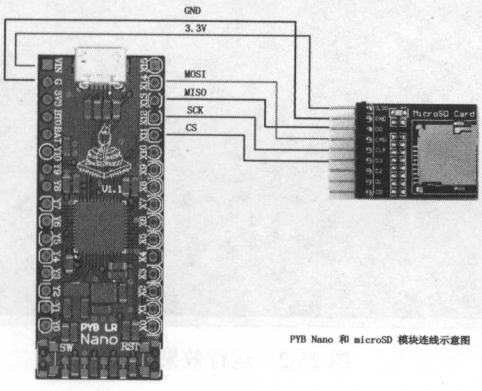


图 25-1　连线图（1）

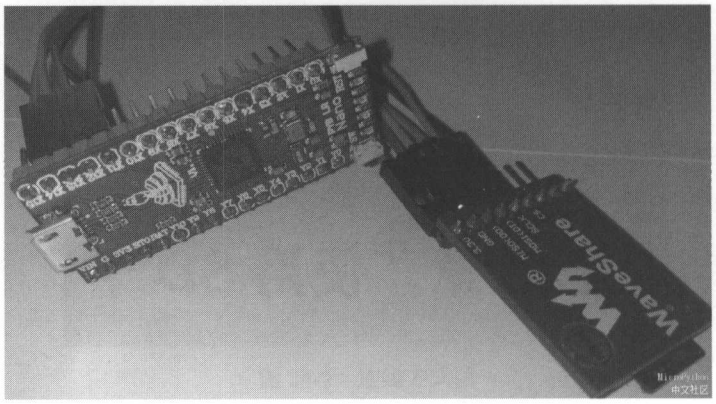


图 25-1　连线图（2）

连接好以后，将 sdcard.py 复制到 PYB mini 中，再使用下面的命令进行挂载，本案例的相关脚本文件名为：NPY_004. py，具体使用的源代码如下。

```
importpyb, sdcard, os

sd = sdcard.SDCard(pyb.SPI(2), pyb.Pin('B12'))
pyb.mount(sd, '/sd2')
os.listdir('/')
```

运行效果图如图 25-2 所示。

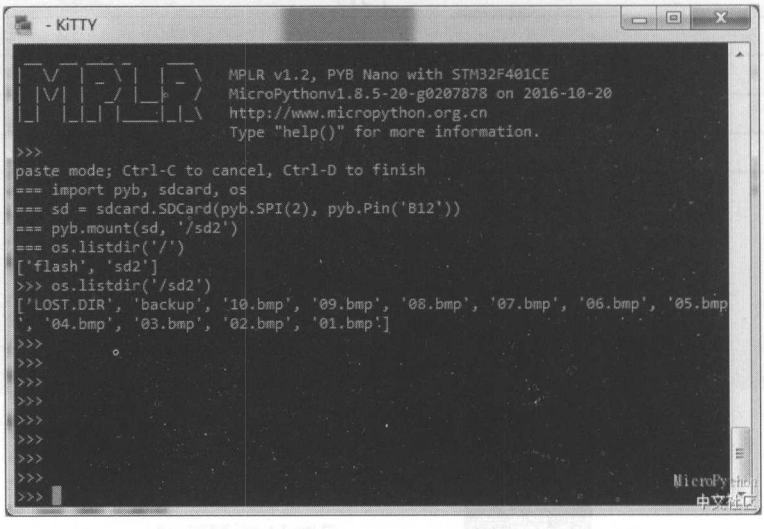


图 25-2　运行效果图

注：

- 上面是连接 SPI2，也可以连接到其他 SPI 上。
- CS 也可以改用其他 GPIO。
- 挂载点必须在根目录，可以用其他名称。
- 图中使用了 WaveShare 的 TF 模块，其他类似的模块也可以。有些模块带有 IDO 和电平转换功能，其 VCC 通常可以接在 5V 上。
- sdcard.py 在 MicroPython 源码的 drivers/sdcard 目录下。

第 26 章　用定位器控制 LED 亮度

使用到一个外部的定位器，并将定位器用杜邦线接到 PYB mini 的引脚：X9/PB1。

本案例的相关脚本文件名为：NPY_005.py，然后输入下面的程序：

```
>>>frompybimportADC, Pin
>>> adc = ADC(Pin('X9'))
>>>while True:
...     dat = adc.read()
...     pyb.LED(1).intensity(dat>>4)
...     print(dat)
...     pyb.delay(200)
```

改变定位器，在屏幕上输出当前 ADC 的结果，同时会改变 LED1 的亮度。

注：

- 上面程序中的 X9 可以改为 X0～X9（支持 ADC 功能）。
- LED（1）可以改为 LED（1）～LED（4），它们都支持亮度调整功能。

第 27 章　计算任意精度的圆周率

计算任意精度的圆周率是一个有趣的主题，得益于 Python 的强大计算能力，我们在 MicroPython 中也可以轻松地计算 pi 的数值。

先创建文件 pi.py，然后输入下面的代码，本案例的相关脚本文件名为：NPY_006a.py。

```
"""
文件：pi.py
说明：用 MicroPython 计算任意精度圆周率计算
作者：未知
版本
时间：
修改：邵子扬
    2016.5 v1.1
    http://bbs.micro-python.com/forum.php
"""
import time

def pi(places=10):
# 3 + 3*(1/24) + 3*(1/24)*(9/80) + 3*(1/24)*(9/80)*(25/168)
# The numerators 1, 9, 25, ... are given by (2x + 1) ^ 2
# The denominators 24, 80, 168 are given by (16x^2 -24x + 8)
  extra = 8
  one = 10 ** (places+extra)
  t, c, n, na, d, da = 3*one, 3*one, 1, 0, 0, 24

while t > 1:
    n, na, d, da = n+na, na+8, d+da, da+32
    t = t * n // d
```

```
        c += t
    return c // (10 ** extra)

    def pi_t(n=10):
        t1=time.ticks_us()
        t=pi(n)
        t2=time.ticks_us()
        print('elapsed: ', time.ticks_diff(t2,t1)/1000000, 's')
        return t

    def pi2(n=10):
        r=6*(10**n)*1000
        p=0
        k=0
        c=r//2
        d=c//(2*k+1)
        while d>0:
            p=p+d
            k=k+1
            k2=2*k
            c=c*(k2-1)//(4*k2)
            d=c//(k2+1)
        return p//1000

    def pi2_t(n=10):
        t1=time.ticks_us()
        t=pi2(n)
        t2=time.ticks_us()
        print('elapsed: ', time.ticks_diff(t2,t1)/1000000, 's')
        return t
```

保存后，将文件 pi.py 复制到 PYB mini 中，然后在终端软件中按下 Ctrl+D 键软件复位，让系统可以识别到 pi.py 文件。然后进行计算测试，同时也可以测试计算速度。（在其他 MicroPython 开发板上也可以进行这个测试），本案例的相关脚本文件名为：NPY_006b.py，本案例的具体代码如下。

```
 __  __ ___ _    ___
|  \/  | _ \ |  | _ \   MPLR v1.2, PYB Nano with STM32F401CE
| |\/| |  _/ |__|   /   MicroPythonv1.8.6 on 2016-11-27
|_|  |_|_| |____|_|_\   http://www.micropython.org.cn
                      Type "help()"for more information.
```

```
>>>import pi
>>> npi=pi.pi_t(1000)
elapsed:  0.714934 s
>>> npi=pi.pi_t(2000)
elapsed:  2.81383 s
>>> npi=pi.pi_t(5000)
elapsed:  16.958 s
>>>
```

从运行结果可以看出，虽然不是很快，但是考虑到 STM32 的资源和性能，结果已经出乎预料了，毕竟计算函数还不到 10 行代码，没有做深度优化。最后将打印出 1000 位的圆周率，大家可以和标准结果比较看看。

```
>>> pi.pi(1000)
3141592653589793238462643383279502884197169399375105820974944592
30781640628620899862803482534211706798214808651328230664709384460955
05822317253594081284811174502841027019385211055596446229489549303819
64428810975665933446128475648233786783165271201909145648566923460348
61045432664821339360726024914127372458700660631558817488152092096282
92540917153643678925903600113305305488204665213841469519415116094330
57270365759591953092186117381932611793105118548074462379962749567351
88575272489122793818301194912983367336244065664308602139494639522473
71907021798609437027705392171762931767523846748184676694051320005681
27145263560827785771342757789609173637178721468440901224953430146549
58537105079227968925892354201995611212902196086403441815981362977477
13099605187072113499999983729780499510597317328160963185950244594553
46908302642522308253344685035261931188171010003137838752886587533208
38142061717766914730359825349042875546873115956286388235378759375195
77818577805321712268066130019278766111959092164201989
>>>
```

注：Pi.py 中包含了两个计算 pi 的函数 pi()和 pi2()，两个函数的功能相同，只是计算公式不同，大家可以运行比较一下，看看运行速度和计算结果有什么不同。

第 28 章　升级固件

在 PyBoard 上使用 MicroPython 时，有时因为各种原因可能需要下载固件到开发板，比如原有程序损坏、固件版本升级、DIY 新的开发板等。下面就介绍升级的方法，也适用于其他使用了 STM32 芯片的 MicroPython 开发板。

在 PyBoard 上，除了可以用 SWD 写入外（开发板并没有设计 SWD 插座，但是都连接到开发板四周的排针上了），更方便的方法是通过 USB 使用 DFU 模式烧写固件。

本处，只介绍在 Windows 系统下的固件升级方法。

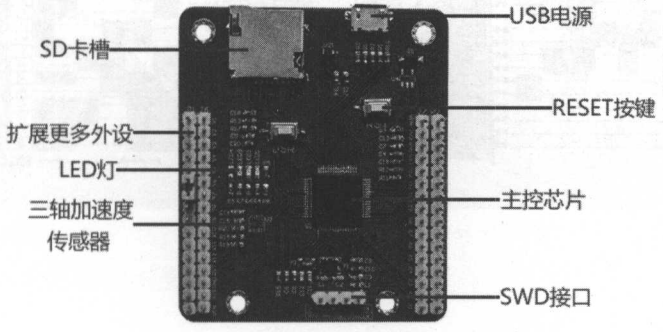
- 首先下载并安装 DfuSedemo，下载地址：http://www.st.com/web/en/catalog/tools/FM147/CL1794/SC961/SS1533/PF257916。
- 运行 DfuSedemo。
- PyBoard 连接 USB。
- 使用终端软件连接到 PYB mini。
- 在 REPL 状态下，输入 pyb.bootloader()，系统将自动进入升级状态。

附录 A　硬件介绍

PyBox 是以遵照 MIT 许可的 MicroPython 为基础，由 TurnipSmart 公司制作的一款 MicroPython 开发板，基于 STM32F405 单片机，通过 USB 接口进行数据传输。该开发板内置 4 个 LED 灯、一个加速传感器，可在 3～10V 之间的电压正常工作。

PyBox 开发板让用户可以通过 Python 代码轻松控制微控制器的各种外设，比如 LED 灯、读取管脚电压、播放歌曲，和其他设备联网等。

PyBox 开发板支持 Python3.0 及以上版本的直接运行，支持重力加速度传感器，支持上百个周边外设配件，支持 SWD 烧写固件。零基础也能灵活掌握单片机技术。如下图所示。

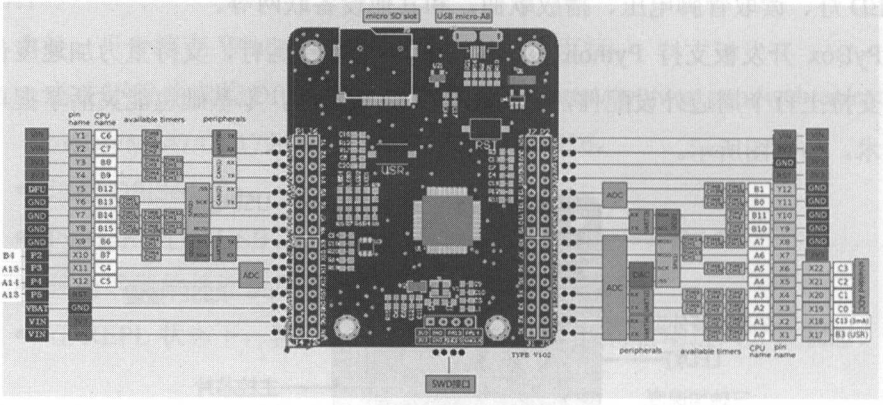


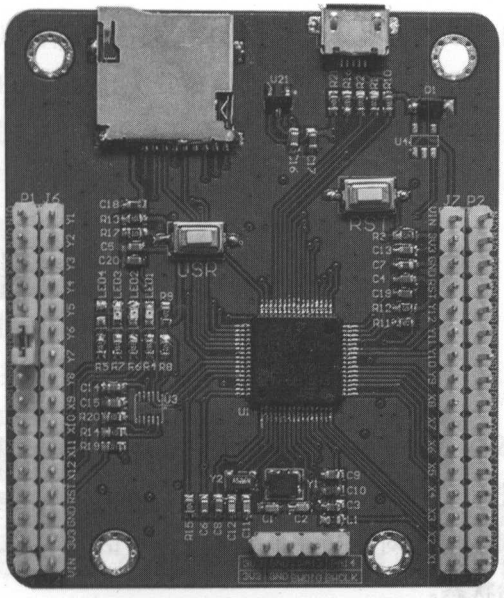
主要特征：

- ARM CORTEX-M4。
- MCU 运行最高速度：168MHz。
- Flash：1024KB。

规格

MCU（主控芯片）	stm32f405rgt6
内存	Flash：1024KB RAM：192KB
存储支持	SD Card（最大 8GB）
电源	Micro USB，可在 3.5V-10V 电压内工作
USB 口	2x USB 2.0
按钮	RST 键：在 Micro USB 口下方 USR 键：RST 键旁边
LED	LED*4（从芯片向外依次是红，黄，绿，蓝）
OS（=固件）	TPYBoardV10X
尺寸	64mm x 54mm
重量	17g
接口	GPIO(30) SPI(2) CAN(2) I2C(2) USART(5) ADC(12) DAC(2) SWD(1)
硬件资源	3 轴加速度传感器（MMA7660） LED*4（从芯片向外依次是红，黄，绿，蓝） TF 卡槽 1 个 按键 2 个




USR
RST

附录 B　安全模式和恢复出厂设置

当你的 PyBox 板除了出了毛病时不用太沮丧，尤其是在编错了程序的时候。首先要做的事情是进入安全模式：这将临时跳过 boot.py 和 main.py 的执行，直接获取默认的 USB 设定。如果对于文件系统有困惑的话可以尝试出厂重置，其将把文件系统恢复为起始状态。

1. 安全模式

按如下步骤操作可以进入安全模式：

1）通过 USB 线连接 PyBox 板并提供电源。

2）按下用户按键。

3）按下用户按键的同时按下重置按键，然后松开。

4）LED 灯将持续亮绿灯，然后橙灯亮再到绿灯、橙灯一起亮的循环。

5）按下用户按键直到橙色的 LED 灯亮起，然后松开用户按键。

6）橙色的 LED 灯将快速闪烁 4 次，然后熄灭。

7）现在你进入了安全模式。

在安全模式下，boot.py 和 main.py 文件将不被执行，因此 PyBox 板将按照默认的设置启动。这意味着现在你可以通过 USB 驱动连接文件系统并对 boot.py 和 main.py 进行编辑以解决问题。

2. 出厂重设置文件系统

如果你的 PyBox 板的文件系统遭到损坏（例如忘记退出或卸载使用），或者你在 boot.py 和 main.py 中编写了无法退出的代码，那么你可以重置文件系统。重

置文件系统将删除开发板里边存储的所有文件（不包括 SD 卡），然后将 boot.py，README.TXT，和 pybcd.inf 文件恢复为其初始状态。

恢复出厂设置的方法与进入安全模式的方法相似，区别只有松开用户按键时亮灯的颜色不同。

8）通过 USB 线连接 PyBox 板并提供电源。

9）按下用户按键。

10）按下用户按键的同时按下重置按键，然后松开。

11）LED 灯将持续亮绿灯，然后橙灯亮，再到绿灯、橙灯一起亮的循环。

12）保持按下用户按键直到橙色和绿色的 LED 灯亮起，然后松开用户按键。

13）绿色和橙色的 LED 灯将快速闪烁 4 次，然后熄灭。

14）红色的 LED 灯亮起（目前红色、绿色、橙色的 LED 灯皆亮）。

15）PyBox 板现在重置了文件系统（这将花费几秒的时间）。

16）所有的 LED 灯一起熄灭。

17）现在你重置了文件系统，并进入了安全模式。

18）按下复位按键后释放将自行启动。

附录 C　使用 Putty 控制 PyBox

在 PyBox 中运行 python 程序，只能直接看到测试结果，比如在第一个例子中点亮 LED 灯，在平常的程序开发中，通常会在编写的程序文件中使用 print 函数打印调试语句，在控制台看到打印结果。那么在 PyBox 中运行 main.py 时有什么办法可以看到调试语句吗？

答案是肯定的，可以使用 Putty 通过串口连接到 PyBox 开发板中。需要进行以下操作：

1）通过 USB 线连接 PyBox 板并提供电源。

2）鼠标右键单击“我的电脑”选择“管理”，打开计算机管理，选择“设备管理器”，如图 C-1 所示。

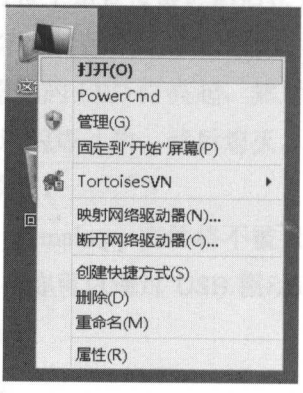


图 C-1

如果在“其他设备”下扫描到的硬件 PyBox 出现黄色的叹号，说明计算机主机没有正确识别 PyBox，需要更新 PyBox 的驱动程序。

读者可以打开 PyBox 所在的盘符，驱动文件在 TPFLASH 的磁盘里，里面的 pybcdc.inf 就是它的驱动文件。

注意：在更新驱动程序前，需要在计算机主机里“**禁用强制驱动签名**”。如果没有执行禁用强制驱动签名的操作就更新驱动程序，在 Windows 8.1 操作系统中将显示“**第三方 INF 不包含数字签名信息**”的出错信息，不能继续安装，如图 C-2 和图 C-3 所示。

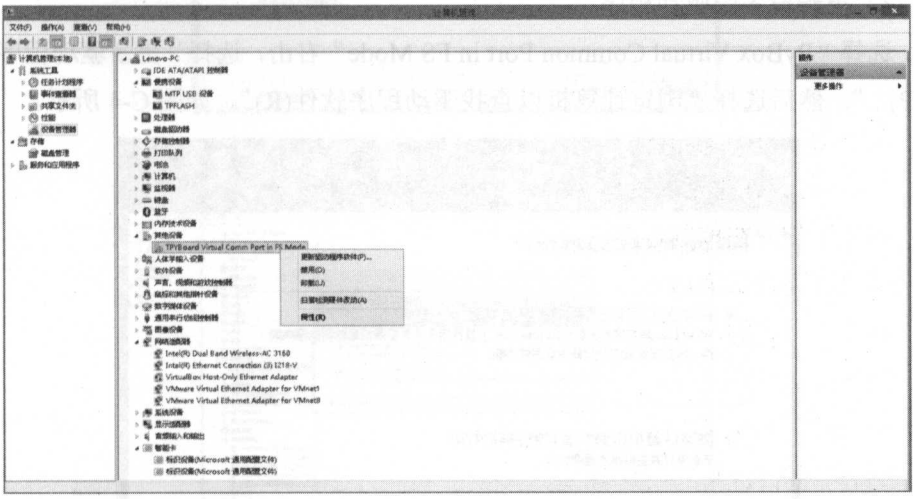


图 C-2

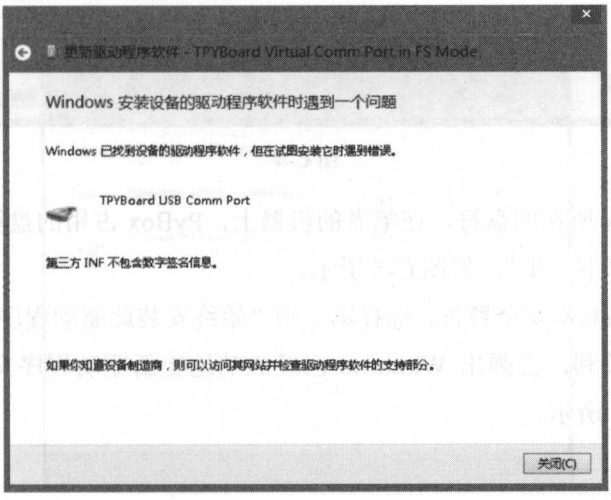


图 C-3

因为数字认证驱动是经过 Windows 实验室（WHQL）认证的驱动程序，在 Windows 8 操作系统更新程序时会自动联网到验证官网查找和设备匹配的驱动程序，而 PyBox 的驱动程序不在其中，所以报错。

在计算机主机里“**禁用强制驱动签名**”具体的操作在以下地址，请读者自行查看。

```
http://jingyan.baidu.com/article/7c6fb42879543380642c9036.html
```

3）更新设备的驱动程序。

选择“PyBox Virtual Common Port in FS Mode”右击，选择”更新驱动程序软件(P)...”，然后选择“浏览计算机以查找驱动程序软件(R)”。如图 C-4 所示。

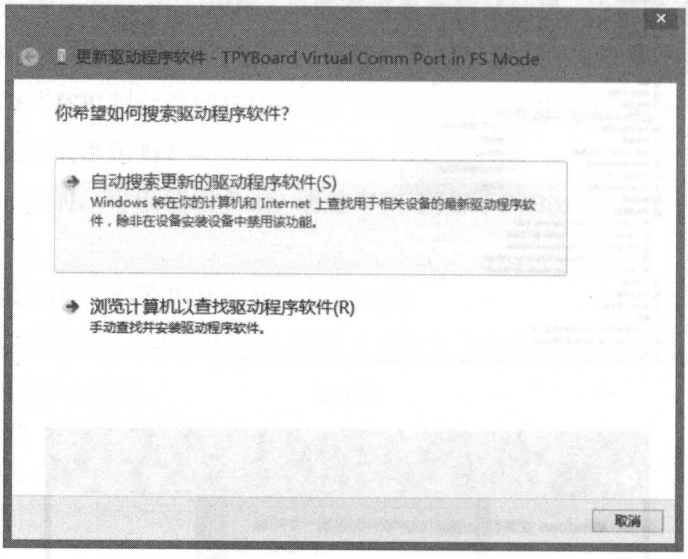


图 C-4

选择 PyBox 所在的盘符，在笔者的机器上，PyBox 占用的盘符在 F：盘。选中后单击按钮“下一步”，如图 C-5 所示。

会弹出 Window 安全警告，选择第二项“始终安装此驱动程序软件(I)”，如果安装驱动一切顺利，会弹出 Windows 已经成功地更新驱动程序文件的提示。如图 C-6 和图 C-7 所示。

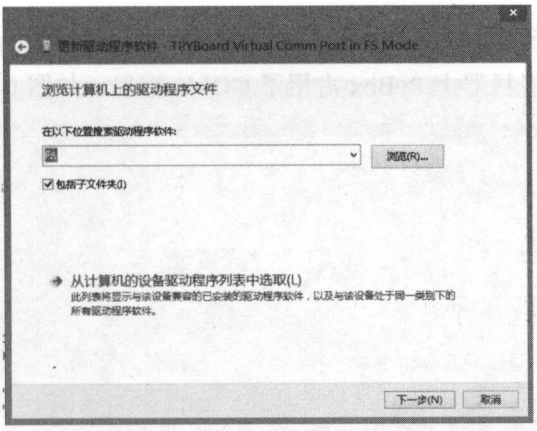

更新驱动程序软件 - TPYBoard Virtual Comm Port in FS Mode
浏览计算机上的驱动程序文件
在以下位置搜索驱动程序软件:
浏览(R)...
包括子文件夹(I)
从计算机的设备驱动程序列表中选取(L)
此列表将显示与该设备兼容的已安装的驱动程序软件，以及与该设备处于同一类别下的所有驱动程序软件。
下一步(N)
取消

图 C-5

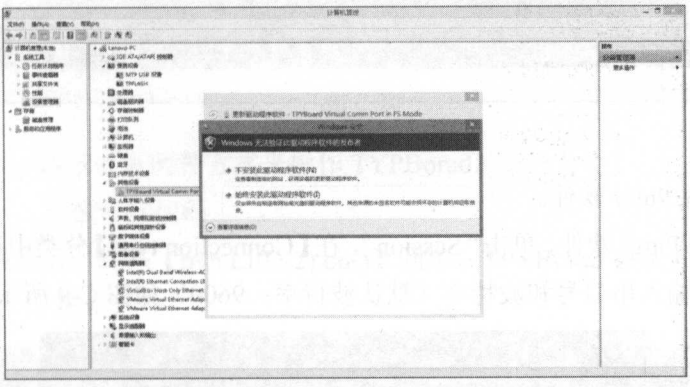


图 C-6

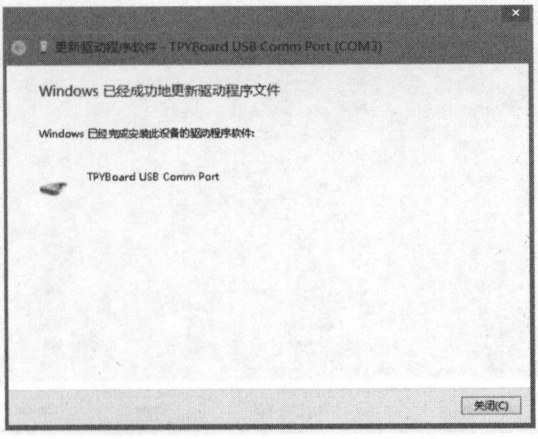

更新驱动程序软件 - TPYBoard USB Comm Port (COM3)
Windows 已经成功地更新驱动程序文件
Windows 已经完成安装此设备的驱动程序软件:
TPYBoard USB Comm Port
关闭(C)

图 C-7

重新扫描硬件改动，会发现在“端口（COM 和 LPT）”下，PyBox 对应的 COM 端口，在笔者机器上 PyBox 占用了 COM3 端口。如图 C-8 所示。

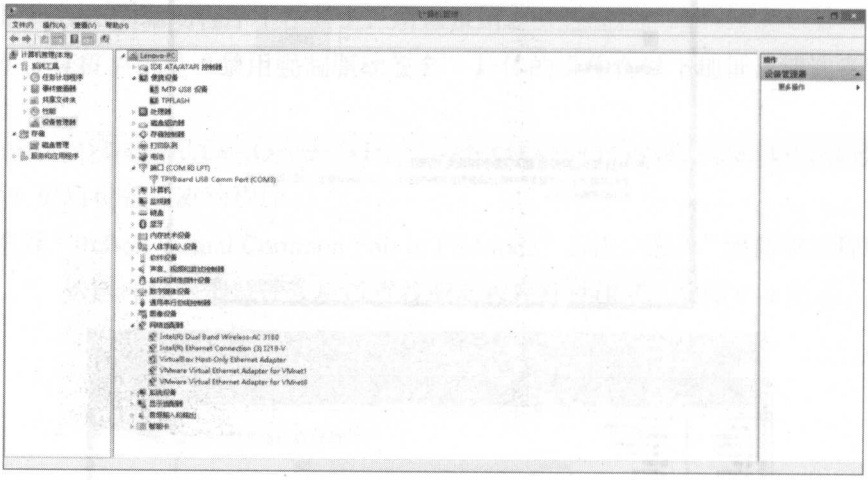


图 C-8

4）下载 Putty 软件。

5）打开 Putty 软件，单击“Session”，在【Connection type】分类中选择【Serial】串口模式，输入串口号和波特率（默认波特率：9600）如图 C-9 所示。

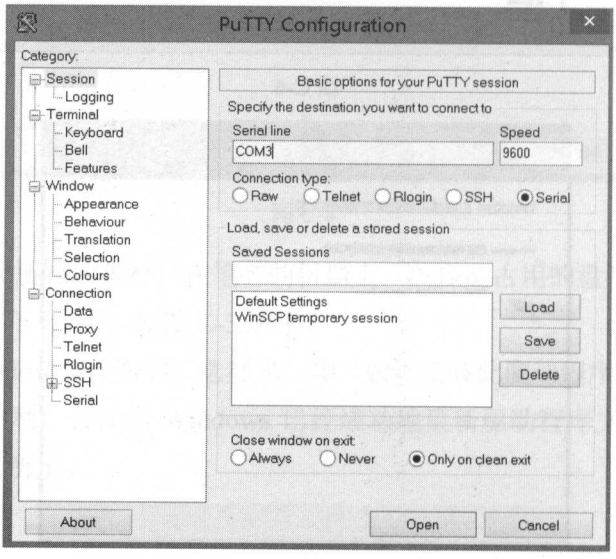


图 C-9

6）单击【Open】，进行连接。连接成功后，如图 C-10 所示。

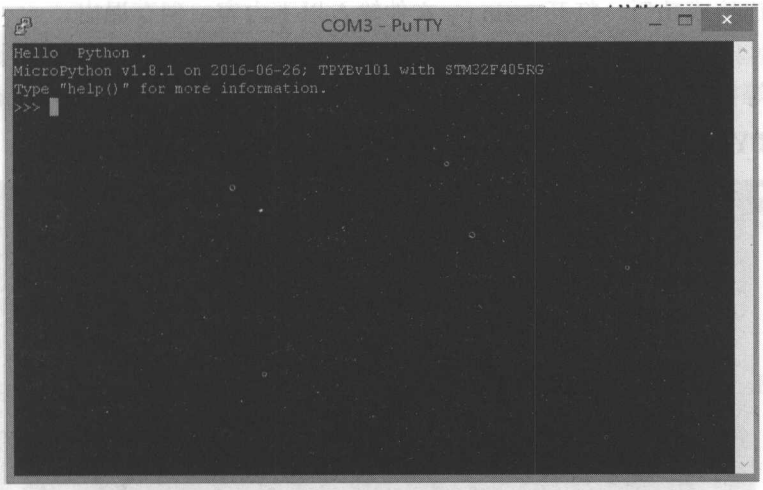


图 C-10

7）Putty 通过以下两种方式来操作 TYPBorad。

方式一：直接运行代码

连接成功后，输入【pyb.LED(2).on()】回车，TPYBorad 板载 LED2 会亮起，效果如图 C-11 所示。

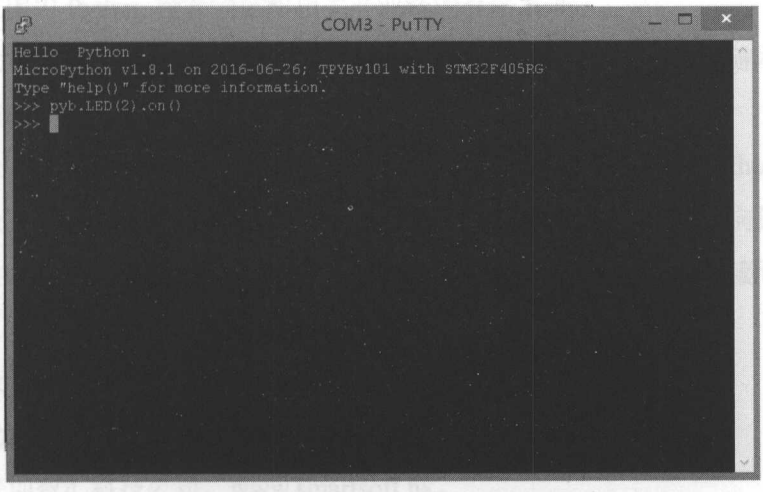


图 C-11

方式二：执行代码文件

① 连接成功后，打开 main.py 文件输入以下内容，保存退出。

```
print('Hello  Python.')
```

② 等待 TPYBorad 板载上的 LED1 熄灭，Putty 输入【execfile(‘main.py’)】回车，TPYBorad 会立即执行 main.py 文件，效果如图 C-12 所示。

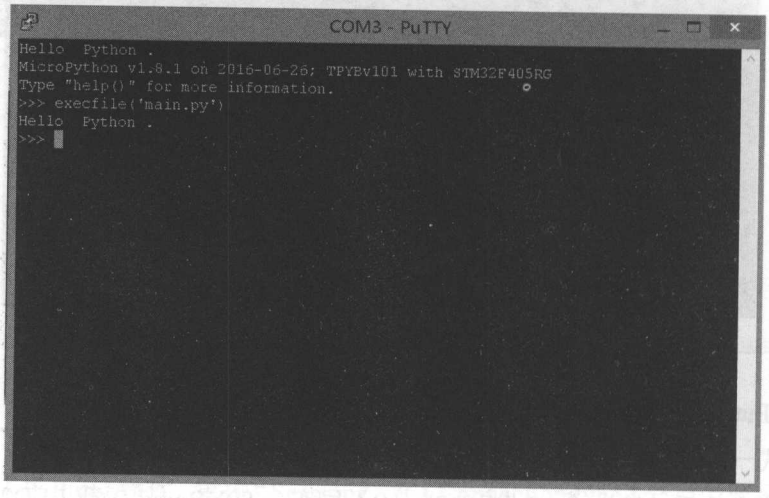


图 C-12

附录 D　Python 极客团队介绍

Python 极客团队（Python Geek Team，简称 PGT），是专注于中国 Python 极客领域、Python 智能硬件领域的开放性技术合作团队。

PGT 团队工作内容包括：

- Python 极客软件、硬件开发，如开发套件、无人机、机器人、人工智能、机器学习、物联网等。
- 跟踪、收集、统计 Python 硬件领域的行业信息和最新科技动态。
- 促进联盟成员在技术、市场、知识产权等领域的交流合作与自律，协同推进国内 Python 极客领域和相关产业链的有序发展。
- 大力推动 Python 智能硬件领域与用户行业之间的深入合作，加速相关技术与产品在各行业中的普及应用。

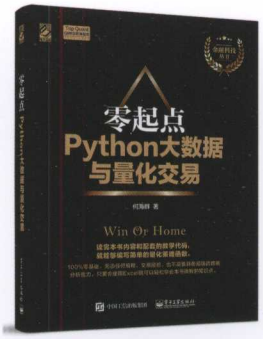
Python 极客团队网址：www.zroboto.com。QQ 群：419523710（python 极客）。

Python 极客团队盟筹备之始，以兼容并蓄、开放合作的姿态，向国内从事 Python 软件、硬件领域的全行业企事业单位，以及民间开发团队发出邀请。有兴趣的企业与团体，请与团队网站负责人联系。

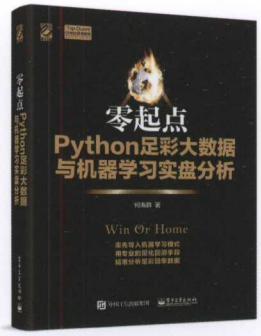
Python 极客团队发起单位包括：

- 北京极宽科技公司，www.TopQuant.vip，www.ziwang.com。
- 深圳智汇科技公司，www.smartsoft.hk。
- 深圳前海智库，www.chrd.cn。

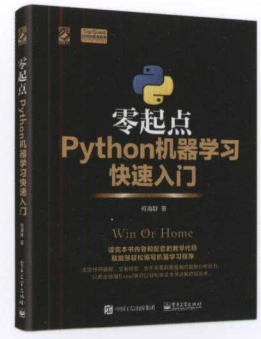
- 山东萝卜电子科技有限公司，www.tpyboard.com，www.turnipsmart.com。
- micropython 中文社区，www.micro-python.com。
- 北京锐和信科技有限公司，www.lapsule.com，www.micropython.cn。
- 深圳壹本堂，www.pkia.cn。



零起点Python大数据与量化交易



零起点Python
足彩大数据与机器学习实盘分析



零起点Python机器学习快速入门

有了先进的软件、硬件开发平台，剩下的只是创意。

中国人，特别是中国的年轻人，是全球最富有创业、创新精神的一群人，这样的一个群体难道还会缺乏创意吗?

《机器人Python极客编程入门与实战》只是“青少年学编程系列丛书”的第一本，本系列包括以下作品。

- 《机器人Python极客编程入门与实战》：Python开发板套件的使用与学习，包括数十个简单入门案例，如LED控制、Wi-Fi控制、机器小车等。
- 《机器人Python智能开发与实战》：基于Python的智能化机器人开发设计，比如语音识别、电脑绘画等。
- 《机器人Python案例汇编》：汇集Python极客团队和国内众多一线高手设计的各种实用、经典智能案例。

“青少年学编程系列丛书”只是“Python极客项目”的起点，也是新一代智能化硬件的起点，我们期待更多的同行、更多的年轻人加入这个领域。

本书所有案例程序均可用于zwPython平台，以及各种支持Python 3的设备平台，包括Linux操作系统、iOS系统，以及安卓系统等。

上架建议：计算机/Python

ISBN 978-7-121-32292-1

9 787121 322921

定价：59.00元

责任编辑：黄爱萍
封面设计：侯士卿